HINDSIGHT PLANNER: A CLOSED-LOOP FEW-SHOT PLANNER FOR EMBODIED INSTRUCTION FOLLOWING

Anonymous authors
Paper under double-blind review

ABSTRACT

This work focuses on building a task planner for Embodied Instruction Following (EIF) using Large Language Models (LLMs). Previous works typically train a planner to imitate expert trajectories, treating this as a supervised task. While these methods achieve competitive performance, they often lack sufficient robustness. When encountering a suboptimal action, the planner may encounter an out-of-distribution state, which can lead to task failure. In contrast, we frame the task as a Partially Observable Markov Decision Process (POMDP) and aim to develop a robust planner under a few-shot assumption. Thus, we propose a closed-loop planner with an adaptation module and a novel hindsight method, aiming to use as much information as possible to assist the planner. Our experiments on the ALFRED dataset indicate that our planner achieves competitive performance under a few-shot assumption. For the first time, our few-shot agent's performance approaches and even surpasses that of the full-shot supervised agent.

1 INTRODUCTION

With the development of AI and robotics, many previous works have combined them to handle Embodied Instruction Following (EIF). Among them, the *Action Learning From Realistic Environments and Directives* (ALFRED) benchmark (Shridhar et al., 2020) is particularly challenging because it requires an agent to learn a long-horizon policy that maps egocentric images and language instructions into a sequence of actions. In each task, the agent will be given a natural instruction (e.g. "Put a heated mug down on a table") and an egocentric visual observation at each step. The agent is required to output low-level actions (e.g. MoveAhead, RotateRight, etc.) based on the observation to complete the task. These tasks are usually challenging due to the sparse reward settings. For such a reason, many works have adopted a hierarchical structure to deal with it (Song et al., 2023; Min et al., 2021; Blukis et al., 2021; Kim et al., 2024). The high-level module decomposes the whole task into several sub-tasks, the low-level module outputs actions to finish each sub-task. Previously, sub-goal planners are trained on human-annotated dataset through supervised learning. However, they require large amounts of data and often lack robustness (Min et al., 2021; Blukis et al., 2021; Kim et al., 2024).

With recent advancements in Large Language Models (LLMs), many studies have explored using LLMs as sub-goal planners, utilizing their in-context learning abilities (Song et al., 2023; Shin et al., 2024; Ahn et al., 2022). Although these methods have achieved competitive performance under the few-shot assumption, a critical limitation is that these approaches all study the problem from a supervised learning perspective. They merely attempt to imitate the ground truth trajectories, which results in a lack of robustness within their agents. EIF benchmarks, on the other hand, require long-horizon planning ability. For example, the task "Put a warmed apple in the fridge" requires 12-step planning. Assuming that after applying in-context learning, the distribution of the agent's output actions becomes closer to that of the Oracle, with an accuracy of 0.9, the overall accuracy of the entire planning task decreases to $0.9^{12} = 0.28$. Traditionally, a large amount of data is required to mitigate such an issue (Blukis et al., 2021; Kim et al., 2024). However, under the few-shot assumption, in-context learning methods rely heavily on the reasoning ability of pretrained LLMs (Brown et al., 2020; Dong et al., 2024). The hallucination problem of LLMs (Zhang et al., 2023) suggests that supervised methods through in-context learning are limited.

To address this issue, we approach the ALFRED task (Shridhar et al., 2020) as a Partially Observable Markov Decision Process (POMDP), where the planner makes decisions based on its current state.

Each task begins with a natural language description. At each step, the planner receives an egocentric RGB image and returns a high-level sub-goal. The planner can only receive reward signals (Success or Fail) at the end of the task. There are three major challenges in building a robust planner: (1) The sparse reward settings make it difficult for the planner to learn and make accurate decisions. (2) The planner can only receive an egocentric picture and cannot detect the whole state. (3) Under the few-shot assumption, the planner cannot obtain enough information from trajectories.

For the first problem, we adopt an actor-critic framework (Liu et al., 2024) which consists of two actors, one critic, and one generator. At each step, the planner receives a new state and performs a tree search with the actors and generator to plan future trajectories, rather than directly outputting a sub-goal. The critic is then used to select the best rollout and return its initial action. Thus, the planner can optimize the output over the long horizon to address the issue of sparse reward. For the second difficulty, we design an adaptation module instantiated by LLMs. Upon receiving an egocentric image, the adaptation module aims to predict the invisible latent PDDL variables of the task, which could help the planner better understand the environment. For the third challenge, we propose a novel hindsight method. It collects suboptimal trajectories from the agent in the training environment and relabels them to complete the task. This approach provides the planner with additional information. During the deployment phase, the relabeled trajectories can guide the planner in adjusting its policy when incorrect actions are proposed and executed.

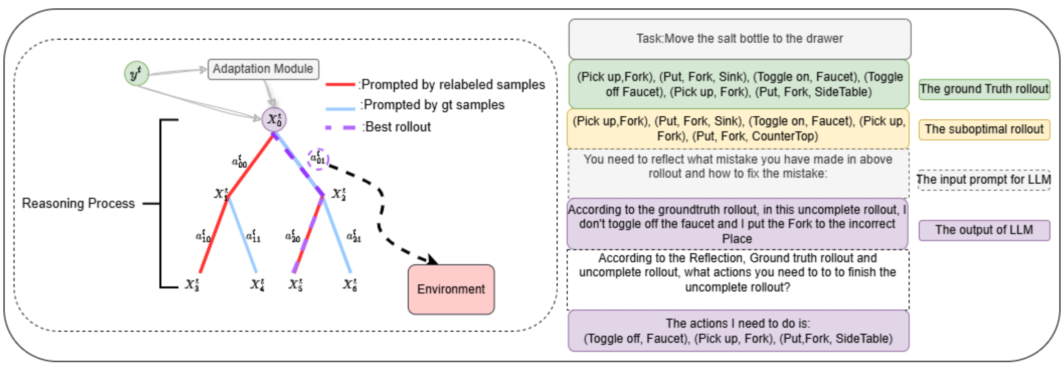

Figure 1: **Left**: The illustration of the Hindsight Planner: at each time step t, the planner receives a partial observation y^t from the environment. The adaptation module estimates the latent variable and concatenates it with y^t to produce the complete state. $\texttt{Actor}_{\texttt{hind}}$ and $\texttt{Actor}_{\texttt{gt}}$ are prompted with different samples and make decisions. The $\texttt{Critic}$ is utilized to evaluate the actions. The best rollout $(x_t, a_t^*, x_{t+1}^*, a_{t+1}^* \ldots)$ is selected, and a_t^* is returned. **Right**: An example of the relabeling process for the $\texttt{Actor}_{\texttt{hind}}$: after collecting a suboptimal rollout, the LLM is prompted to generate a reflection on the previously taken actions. Following this reflection, the LLM is then prompted to complete the suboptimal rollout.

In summary, our contributions are threefold:

(1) We study ALFRED (Shridhar et al., 2020) from a POMDP perspective for the first time and propose a closed-loop actor-critic planner to solve it.

(2) We propose a novel hindsight prompting method and demonstrate that our method is theoretically superior to previous approaches.

(3) Experiments on ALFRED (Shridhar et al., 2020) show that our method achieves state-of-the-art performance under few-shot assumptions. Specifically, the success rates for the "Test Seen" and "Test Unseen" splits are 25.51 and 18.77, respectively, representing a 60% and 39% improvement over the previous baseline.

2 Related Work

2.1 Large Language Model (LLM) and In-Context Learning (ICL)

Large language models (LLMs) have shown incredible reasoning ability (Vaswani et al., 2023; Wei et al., 2022; Touvron et al., 2023; OpenAI et al., 2024) across a wide range of tasks. A crucial way to enhance this reasoning ability is through in-context learning (ICL) (Brown et al., 2020; Dong et al., 2024), which allows LLMs to solve complex tasks with only a few samples. Furthermore, this approach removes the need for fine-tuning, which can be time-consuming and computationally expensive. To utilize the ICL ability better, many studies propose certain frameworks aimed at enhancing the reasoning capabilities of LLMs (Yao et al., 2023; Wei et al., 2023; Yao et al., 2024). Among them, Liu et al. (2024) proposes a novel perspective by bridging RL and LLM, which inspires us to study ICL from an RL aspect. Xie et al. (2022) interprets ICL as Implicit Bayesian Inference, while Dai et al. (2023) believes that ICL is performing implicit Gradient Descent. All of these imply the importance of the content in ICL, an area that remains relatively understudied. To this end, we propose Hindsight Planner as an exploration.

2.2 Adaptation Module in POMDP

In a Partially Observable Markov Decision Process (POMDP), planners are presented with observable states, while the latent states are invisible to the planner. Making decisions with incomplete information is challenging; therefore, a component to map the observable state into the latent space is crucial (Lee et al., 2023). Adaptation modules have been proven effective in legged robots (Kumar et al., 2021; Zhou et al., 2019; Peng et al., 2020). These modules aim to bridge the gap between the simulator and the real world. They are often trained to predict crucial information that a robot can sense in the simulator but not through its sensors in the actual world, such as surface friction or payload of the robot. The base policy then makes decisions based on the observed information and the invisible latent information predicted by adaptation modules. Inspired by this, we propose an adaptation model that maps the visible object list to the latent, invisible Planning Domain Definition Language (PDDL) (Chapman, 1987) of ALFRED (Shridhar et al., 2020).

Previous work such as Min et al. (2021), trains a BERT (Devlin et al., 2019) to predict the PDDL arguments and decompose high-level instructions into templated sub-goals. However, our approach differs from these in two aspects: (1) Previous works predict the arguments at the beginning of a task, which is equivalent to predicting the latent variables based on the initial observed state. In contrast, our method predicts the latent arguments at each time before reasoning, allowing predictions to be adjusted through exploration, which makes our planner more robust. (2) We do not apply the templated approach directly. The adaptation module is used to reveal the latent information for the planner and assist the planner in making better decisions. Experiments show that our method achieves competitive performance even without the assistance of the adaptation model, as demonstrated in Table 4.

2.3 Hindsight in LLMs

Hindsight algorithms (Andrychowicz et al., 2018; Li et al., 2020; Pong et al., 2020) are widely adopted in the reinforcement learning (RL) area. Generally, the hindsight method aims to reveal future information after collecting a trajectory and relabel the trajectory to make it more informative during training process (Furuta et al., 2022; Andrychowicz et al., 2018). Furuta et al. (2022) applies the hindsight method in training a Transformer model and achieves competitive performance on several baselines. However, training a model from scratch usually requires a large amount of data. In contrast, in-context learning, leveraging the reasoning ability of LLMs, allows an agent to complete complex tasks with only a few samples. Dai et al. (2023) has shown that ICL executes an implicit parameter update. As a result, we utilize ICL in our proposed method. Intuitively, we hope hindsight prompts can provide guidance when an out-of-distribution state is encountered. For example, “Wash a pan and put it away” requires the agent to wash a *Pan* and put it on the *DiningTable*. The trajectory from a planner could be: {(PickupObject, Pan), (PutObject, Sink), (ToggleObjectOn, Faucet), (PickupObject, Pan), (PutObject, CoffeeMachine)}. Note that in this example, the agent fails to place the pan in the correct location, does not turn off the faucet, and thus the trajectory from the planner is suboptimal. Our hindsight method proposes a novel relabeling process that appends actions to the suboptimal trajectory, aiming to complete the task. In the above example, the corrected trajectory should be: {(PickupObject, Pan), (PutObject, Sink), (ToggleObjectOn, Faucet), (PickupObject, Pan), (PutObject, CoffeeMachine), (ToggleObjectOff, Faucet), (PickupObject,Pan), (PutObject, DiningTable)}. This approach enables us to guide the planner in addressing unknown states resulting from incorrect actions. Consequently, during the deployment phase, when the planner encounters a similar state, it can learn from suboptimal trajectories and subsequently take correct actions to correct previous mistakes.

We also analyze our method in comparison to previous hindsight methods (Andrychowicz et al., 2018; Ghosh et al., 2019) following the framework proposed by Furuta et al. (2022). We demonstrate that while previous methods are effective, they alter the distribution of a crucial variable in multi-task RL problems. In contrast, our method optimize the same objective while maintaining the distribution. The detailed discussion can be found in Section 4.3.

3 PRELIMINARIES

3.1 DEFINITION IN POMDP

In a POMDP $\mathcal{M}$, consider an action space $\mathcal{A}$, latent state space $\mathcal{X}$, observation space $\mathcal{Y}$, transition probability function $p(x'|x,a)$, emission function $o(y|x)$, reward function $r(x,a)$ and discount factor $\gamma \in [0,1)$. The policy $\pi_\theta(\cdot|y)$ maps the latent state space to the action space, where θ represents its parameter. The goal of RL is to train a policy such that

$$\pi_\theta = \arg\max_\pi \frac{1}{1-\gamma}\mathbb{E}_{x\sim\rho^\pi(x),y\sim o(\cdot|x),a\sim\pi(\cdot|y)}[r(x,a)], \tag{3.1}$$

where $\rho^\pi(x) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t p_t^\pi(x_t = x),$

$$p_t^\pi(x) = \int_{x_{0:t},y_{0:t-1},a_{0:t-1}} \prod_{i=1}^{t} p(x_i|x_{i-1},a_{i-1})o(y_{i-1}|x_{i-1})\pi(a_{i-1}|y_{i-1}).$$

Given a parameterized reward function $r_z(x,a)$, $z \in \mathcal{Z}$ is a variable that indicates the goal for the agent. A conditional policy $\pi(a|y,z)$ is trained with different values of z where $z \sim p(z)$. The goal in Equation (3.1) becomes

$$\pi_\theta = \arg\max_\pi \frac{1}{1-\gamma}\mathbb{E}_{z\sim p(z),x\sim\rho_z^\pi(x),y\sim o_z(\cdot|x),a\sim\pi(\cdot|y,z)}[r_z(x,a)]. \tag{3.2}$$

Equation (3.2) can be considered as the multi-task RL objective to optimize, which is the core of EIF.

3.2 INFORMATION MATCHING

Following Furuta et al. (2022), we define the information matching (IM) problem as training a policy π_θ that satisfies

$$\pi_\theta = \arg\min_{\pi_\theta} \mathbb{E}_{z\sim p(z),\tau\sim\rho_z^\pi(\tau)}\left[D(I(\tau),z)\right], \tag{3.3}$$

where $I(\tau)$ is *information statistic* that can be any function that captures the desired information from a partially observed trajectory $\tau_t = \{y_0, a_0, y_1, y_1, \ldots, y_t\}$ and D is a divergence measure such as Kullback-Leibler (KL) divergence or some f-divergences. Competitive results have been achieved with this optimization objective (Lee et al., 2020; Hazan et al., 2019).

Furuta et al. (2022) demonstrates that previous hindsight methods (Andrychowicz et al., 2018; Eysenbach et al., 2020; Guo et al., 2021) utilize various *information statistics* and minimize the divergence $D = 0$ by setting $\widehat{z} = I(\tau)$. This allows trajectories to be better used to train a policy $\pi(a|s,z)$. For instance, in HER (Andrychowicz et al., 2018), an MDP trajectory $\tau_t^s = \{s_0, a_0, s_1, \ldots, s_t\}$ is collected. The information statistic is set as the final state of the agent, where $I(\tau_t^s) = s_t$, and the relabeling process in HER is equivalent to setting $\widehat{z} = I(\tau_t^s)$.

4 HINDSIGHT PLANNER

4.1 OVERVIEW

The Hindsight Planner outputs a sub-goal based on the observed objects and natural language instructions. During the collection phase, suboptimal trajectories are collected, and we apply our

Algorithm 1 Hindsight Planner

Input: An LLM-planner `LLM-PL`, an adaptation module `Adapter` and the task instruction I.
Set: Observed Objects $O \leftarrow \varnothing$, the sub-goal history $G \leftarrow \varnothing$, the current sub-goal $S \leftarrow \varnothing$, the time step $t \leftarrow 0$ and sub-goal index $k \leftarrow 0$.
Get sample pool $\mathcal{D}$ and initialize $\texttt{Actor}_\theta$, `Critic`, `Adapter` from $\mathcal{D}$, for any $\theta \in \{\texttt{gt}, \texttt{hind}\}$ (e.g. Algorithm 3 in Appendix A). (Hindsight process)
while Not *Finished* **do**
  Get PDDL arguments $P \leftarrow \texttt{Adapter}(I, O)$.
  Plan and get sub-goal $S_k \leftarrow \texttt{LLM-PL}(\texttt{Actor}_{\texttt{gt}}, \texttt{Actor}_{\texttt{hind}}, \texttt{Critic}, P, I, O, G)$(e.g. Algorithm 2 in Appendix A).
  Set S_k as sub-goal for `Low-PL`.
  while S_k not *Finished* and not *Failed* **do**
    Invoke `Low-PL` to plan and execute a_t and update O.
    Set $t \leftarrow t + 1$
    if S_k *Finished* **then**
      Append S_k to G.
      Set $k \leftarrow k + 1$.
    end if
  end while
end while

hindsight method to generate $\mathcal{D}_{hind}$. The complete dataset $\mathcal{D} = \mathcal{D}_{\text{hind}} \cup \mathcal{D}_{\text{gt}}$, where $\mathcal{D}_{\text{gt}}$ is constructed from training data. In the deployment phase, we initiate hindsight actor $\texttt{Actor}_{\texttt{hind}}$, ground truth actor $\texttt{Actor}_{\texttt{gt}}$, and `Critic` from $\mathcal{D}$.

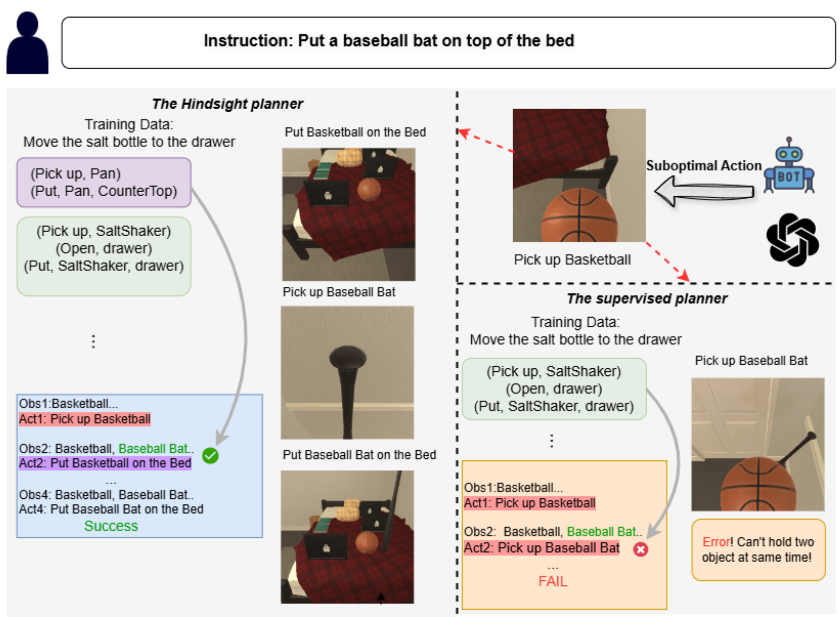

Figure 2: A comparison of Hindsight Planner and previous supervised methods when taking a suboptimal action. The agent initially picks up the incorrect object ("Basketball"). In the supervised method, the planner fails to handle this situation, which leads to task failure. In contrast, the Hindsight Planner can adjust after the incorrect action and successfully complete the task.

At time step t, the planner receives an observed object list y_t from observation functions (Blukis et al., 2021). We then apply the Adaptation module to predict the latent PDDL arguments P based on y_t. The whole state x_t is constructed by y_t and P. With x_t, we invoke the actor-critic task planner `LLM-PL` to generate a future trajectory over a long horizon and return the sub-goal S_k. To ensure the output from the planner meets the requirements, a frozen BERT (Devlin et al., 2019) is used to map the output to the legal space. The proposed sub-goal will be executed by a low-level controller `Low-PL` (Blukis et al., 2021). When a sub-goal is completed or fails, the planner reinvokes the reasoning process to replan another future trajectory from the new state. The complete algorithm is presented in Algorithm 1, and Figure 3 provides an example of the entire process.

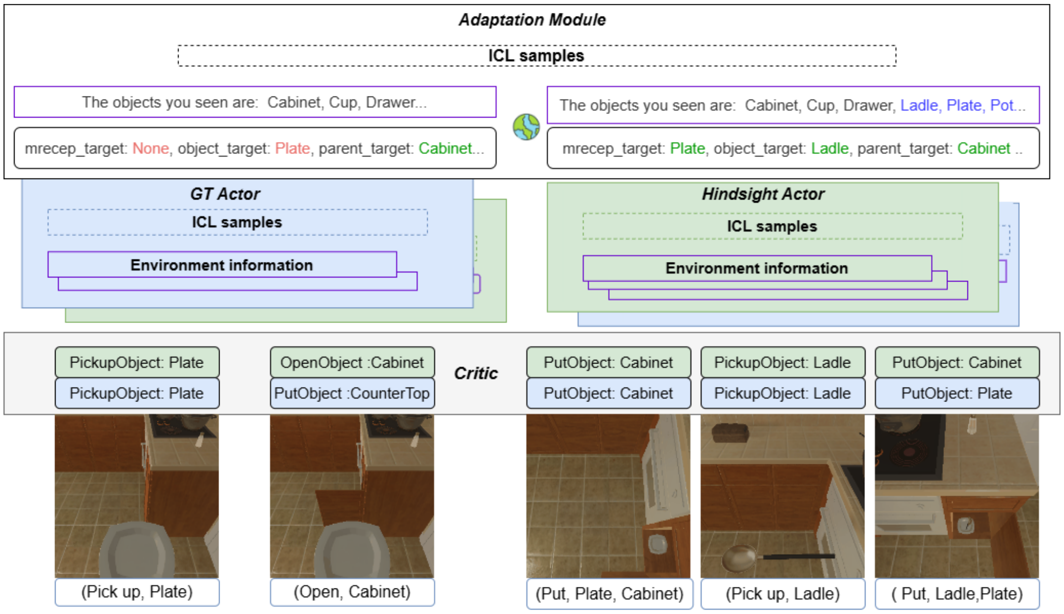

Figure 3: The entire process of the Hindsight Planner is as follows: At the start of the task, which is to "Place a plate with a ladle on it in a cabinet," the `Adapter` mistakenly identifies the task as picking up a plate and placing it into a cabinet. $\texttt{Actor}_{\texttt{hind}}$ and $\texttt{Actor}_{\texttt{gt}}$ make decisions separately. `Critic` then selects the best action as its output. Upon further exploration, the agent detects more objects, and the `Adapter` adjusts its output, recognizing the task as stacking a ladle onto a plate and then placing them into a cabinet. The `Actors` and `Critic` subsequently make decisions based on the revised predictions.

4.2 Prompt Design

All components follow a similar design. The prompt begins with an intuitive explanation of the task and a role description of the LLM. A frozen BERT is then used as a kNN retriever, encoding the task description and selecting K examples with the closest Euclidean distance from the sample pool as in-context samples (Song et al., 2023). Intuitively, the planner would make similar suboptimal actions in similar tasks. For instance, if in an in-context sample "Place two spray bottles into the cabinet," the planner fails to open the cabinet when putting the second spray bottle into it. In the current task "Putting two candles in a cabinet", the planner would know to avoid a similar mistake. The detailed prompts for each process can be viewed in Appendix B.

4.3 Hindsight Method

In Section 3, we gain a coherent framework to describe previous hindsight methods. However, we find that such methods can lead to the policy π being suboptimal, particularly when the number of samples is insufficient. To illustrate this better, we consider the optimization objective in Equation (3.2). It aims to learn a policy under different values of z where $z \sim p(z)$. During the collection phase, the agent's trajectory is usually suboptimal and random. Assume the distribution of $I(\tau) \sim q$. The training objective after relabeling is to train a policy $\widehat{\pi}$ satisfies that

$$\bar{\pi} = \arg\max_{\pi} \frac{1}{1-\gamma} \mathbb{E}_{z\sim q(z), x\sim\rho_z^{\pi}(x), y\sim o(\cdot|x), a\sim\pi(\cdot|y,z)} \left[r_z(x,a)\right]. \tag{4.1}$$

Define the π^* as the oracle. It is easy to see that

$$\begin{aligned} &\frac{1}{1-\gamma} \mathbb{E}_{z\sim p(z), x\sim\rho_z^{\bar{\pi}}(x), y\sim o(\cdot|x), a\sim\bar{\pi}(\cdot|x,z)} \left[r_z(x,a)\right] \\ <&\frac{1}{1-\gamma} \mathbb{E}_{z\sim p(z), x\sim\rho_z^{\pi^*}(x), y\sim o(\cdot|x), a\sim\pi^*(\cdot|x,z)} \left[r_z(x,a)\right], \end{aligned} \tag{4.2}$$

as the distribution of z is shifted from p to q.

Based on such discovery, we propose a new method of hindsight. Assume that τ^* is the ground truth rollout from the oracle π^*, we can rewrite $z = I(\gamma^*)$, Equation (3.3) then becomes

$$\min_{\pi} \mathbb{E}_{z\sim p(z),\tau\sim\rho_z^{\pi}(\tau)} \left[D(I(\tau), I(\tau^*))\right]. \tag{4.3}$$

Our method utilizes LLMs to relabel $\widehat{\tau} = \tau_T + \{a_T, y_{T+1}, a_{T+2}, \ldots\}$ in such a way that $I(\widehat{\tau}) = I(\tau^*) = z$. Thus, we minimize the divergence in Equation (4.3) while keeping the distribution of z unshifted. Intuitively, Equation (4.2) shows that relabeling z alters the distribution of tasks that are truly relevant to our daily lives. This is especially crucial in the reasoning process of EIF.

In practice, our hindsight method consists of two main parts: the collection phase and the deployment phase. During the collection phase, the planner executes tasks and retrieves K examples from a small set of ground truth samples. At each task, the planner generates a possibly suboptimal trajectory τ and relabels them. The algorithm is summarized in Algorithm 3 of Appendix A. During the deployment phase, the $\texttt{Actor}_{\texttt{gt}}$ is prompted with ground truth samples while the $\texttt{Actor}_{\texttt{hind}}$ and the `Critic` are prompted with relabeled samples. Intuitively, we hope that the $\texttt{Actor}_{\texttt{gt}}$ can provide the correct action to complete the task along the shortest path. However, when an incorrect action—which is often unavoidable—is executed, the $\texttt{Actor}_{\texttt{hind}}$ and the `Critic` should be able to correct it. The relabeling process utilizes the reasoning ability of LLMs to fit suboptimal trajectories into correct rollouts. The CoT (Wei et al., 2023) method is utilized in the relabeling process. We first prompt the LLM to generate a *Think* about the suboptimal rollout and then prompt it to complete the suboptimal rollout based on the *Think*. A comparison of the hindsight method with the supervised methods is shown in Figure 2, while the right half of Figure 1 illustrates an example of the relabeling process.

4.4 Adaptation Module

In a POMDP, the adaptation module is used to predict the latent variables from the observed environment y_t (Lee et al., 2023; Kumar et al., 2021) and construct the whole state $x_t = (\texttt{Adapter}(y_t), y_t)$. In practice, we utilize an LLM as the adaptation module and set PDDL arguments as the prediction target for it. The input prompt for the adaptation module begins with an intuitive explanation of ALFRED, followed by several in-context samples. At the end of the prompt is the current task and the object list. At each step, the object list is updated as the agent explores the environment.

The output from the adaptation module varies depending on the task description. Inspired by PDDL (Chapman, 1987; Silver et al., 2023) of ALFRED, the adaptation module needs to predict the following arguments at each step: (1) *object_target*: The specific object to be interacted with during the task. (2) *parent_target*: The final place for the object in the task. (3) *mrecep_target*: The container or vessel necessary for the task. (4) *toggle_target*: The device that needs to be toggled in the task. (5) *object_state*: Indicates whether the target object needs to be cleaned, heated, or cooled. (6) *object_sliced*: Determines if the object must be sliced. (7) *two_object*: Specifies whether the task involves handling and placing two objects. The adaptation module predicts these arguments at each time before reasoning. Then, the arguments are processed into a specific format to assist the task planner to sense the environment better.

4.5 Task Planner

We adopt an actor-critic planner (Liu et al., 2024). At each time step t, the planner receives x_t from the environment and the adaptation module. We initiate two Actors: $\texttt{Actor}_{\texttt{gt}}$ and $\texttt{Actor}_{\texttt{hind}}$, with different samples from the sample pool $\mathcal{D}$. For each state, we prompt each Actor to generate $\frac{W}{2}$ actions. The `Critic` then selects the top B actions. A generator ψ generates the next state based on each action. In this way, we map `Actors` and `Critic` to B future trajectories and select the best future trajectory $(x_t, a_t^*, \ldots)$ through `Critic`. a_t^* is then returned as the sub-goal for `Low-PL`. The left half of Figure 1 shows the reasoning process of the planner.

5 Experiment

5.1 Setups

We validate our framework using the ALFRED benchmark (Shridhar et al., 2020). This benchmark assesses the agent's capability to execute a series of actions for long-horizon household tasks based

Model	n-shot	Test Seen SR	Test Seen GC	Test Unseen SR	Test Unseen GC
HiTUT (Zhang & Chai, 2021)	full	13.63	21.11	11.12	17.89
HLSM (Blukis et al., 2021)	full	25.11	35.79	20.27	27.24
FILM (Min et al., 2021)	full	28.83	**39.55**	**27.80**	**38.52**
MCR-Agent (Bhambri et al., 2024)	full	**30.13**	-	17.04	-
FILM (low inst.) (Min et al., 2021)	few	0.00	4.23	0.20	6.71
LLM-Planner (Song et al., 2023)	few	15.33	24.57	13.41	22.8
LLM-Planner (low inst.) (Song et al., 2023)	few	18.80	26.77	16.42	23.37
Socratic-Planner(Shin et al., 2024)	few	13.24	21.51	10.66	19.53
Hindsight planner (ours)	few	**25.51**	**34.74**	**18.77**	**28.29**

Table 1: **Comparison with the state-of-the-art methods on SR and GC in the test set.** Bold symbols in numbers denote the highest accuracy, while underlined symbols indicate the runner-up for each experiment setting. "low inst." refers to the use of step-by-step instructions.

on natural language task descriptions and egocentric vision. The ALFRED dataset consists of 25k annotations, 108 distinct objects, 7 types of tasks, and 120 scenes. The dataset is divided into training, validation, and testing splits. The validation and test splits contain "seen" subsets, which are part of the training fold, and "unseen" subsets, which are distinct from it. The evaluation is based on Success Rate (SR) and Goal Condition (GC). Given the inherent noise in natural language instructions and the complexities of long-horizon task planning, the ALFRED benchmark presents significant challenges for embodied agents in formulating robust and precise plans.

Similar to previous work (Song et al., 2023; Shin et al., 2024), we only utilize a few examples from the 21k training set annotations. For each of the 7 task types, we randomly select 20 trajectories as the initial sample pool. At the collection phase, we run our planner on the 140 trajectories and collect sub-optimal trajectories. During collection, the same task is not included as in-context samples.

We then give a detailed discussion of the relabeling process. Directly applying the task description from ALFRED may lead to unsatisfactory results, as the task description is often vague. For example, the task "Put a chilled potato on the small black table" requires the planner to put the *potato* on a *SideTable*. If the task description is applied directly, LLMs might focus incorrectly on the *Black Table* and return an incorrect action "PutObject BlackTable". If the task description is not included in the prompt, it could lead LLMs to imitate the ground truth trajectory. However, planners usually have multiple ways to complete a certain task. For instance, in a task requiring the planner to slice an apple, after slicing the apple, the planner could put the *Knife* on the *DiningTable* or *CounterTop*. To address this issue, we relabel the task based on the latent PDDL arguments. The task description "Put a chilled potato on the small black table" becomes "Pick up one cooled potato and put it on the SideTable". This approach helps clarify the task for the planner and reduces the ambiguity in instructions.

For the kNN retriever, we use a frozen BERT from Wolf et al. (2020). We employ GPT-4 Turbo as the target LLM and set temperature to 0. For the `Adapter`, 5 in-context examples are retrieved from the sample pool through the kNN retriever. For the `Actors` and `Critic` modules, 2 in-context examples are retrieved. The task planner uses beam search with a depth and width of 2. To preserve the few-shot assumption and ensure a fair comparison, we directly adopt the pretrained modules for navigation, perception, and low-level control from HLSM (Blukis et al., 2021).

5.2 Main Results

We initially compare our method to other few-shot methods, as shown in Table 1. It is evident that our method achieves a 10.18 and 5.36 higher success rate in "Test Seen" and "Test Unseen" categories, respectively, compared to the previous state-of-the-art method (LLM-Planner) that uses high-level instructions only. Moreover, even when compared to methods utilizing low-level, step-by-step instructions, our method still demonstrates superior performance.

We also compare our method to the other approaches under the same low-level controller (Blukis et al., 2021) in Table 2. The results indicate that our method not only significantly outperforms

Model	n-shot	Valid Seen SR	Valid Seen GC	Valid Unseen SR	Valid Unseen GC	Test Seen SR	Test Seen GC	Test Unseen SR	Test Unseen GC
HLSM (Blukis et al., 2021)	full	**29.63**	**38.74**	18.28	**31.24**	25.11	**35.79**	**20.27**	27.24
LLM-Planner (Song et al., 2023)	few	13.53	28.28	12.92	25.35	15.33	24.57	13.41	22.8
Socratic-Planner (Shin et al., 2024)	few	14.88	25.47	13.40	24.91	13.24	21.51	10.66	19.53
Hindsight planner (ours)	few	25.61	34.95	**19.00**	29.90	**25.51**	34.74	18.77	**28.29**

Table 2: **Comparison with the same lower-controller.** Bold symbols in numbers denote the highest accuracy, while underlined symbols indicate the runner-up for each experiment setting.

Task Type	Examine	Pick	Clean	Stack	Pick Two	Heat	Cool
Base Method	40.42	50	15.18	9.56	30.65	7.48	21.43
w.o. hindsight prompt	39.36	49.29	16.96	7.82	29.84	7.47	10.31
w.o. adaptation module	35.1	47.1	8.93	6.09	32.25	9.34	18.26

Table 3: Ablation study on the success rate of different type of tasks in "Valid Seen" split.

previous few-shot LLM planners but also, for the first time, a few-shot LLM method (with around 100 examples) nearly matches and even surpasses (SR in "Valid Unseen", "Test Seen", and GC in "Test Unseen") fully supervised (around 21k samples) methods.

5.3 Ablation Study

We conduct ablation studies to understand the effectiveness of the components in our framework. First, we ablate the adaptation module `Adapter`, which requires the planner to make decisions based solely on the partially observed information. The results show that this causes a drop of -2.44 and -4.01 in the success rates for the "Valid Seen" and "Valid Unseen" splits. Then, we remove the hindsight prompts. For a fair comparison, the original planner requires both $\texttt{Actor}_{\texttt{gt}}$ and $\texttt{Actor}_{\texttt{hind}}$ to generate one action per state. We also ablate by prompting $\texttt{Actor}_{\texttt{gt}}$ to output two actions for each state. Table 4 shows that the success rates drop by -2.08 and -2.68 in the "Valid Seen" and "Valid Unseen" splits.

Model	Valid Seen SR	Valid Seen GC	Valid Unseen SR	Valid Unseen GC
W.O. Adaptation module	23.17	33.28	14.99	27.36
W.O. Hindsight Prompt	23.53	32.76	16.32	28.06
Base Method	25.61	34.95	19.00	29.90

Table 4: Ablation on "Valid Seen", "Valid Unseen" splits.

For a more comprehensive analysis, we report the success rates for each task type in the "Valid Seen" split, as shown in Table 3. Concurrently, we also present the average sub-goal lengths in the right table. This analysis reveals that hindsight prompting is especially crucial in relatively long-horizon tasks, such as "Cool Object" and "Heat Object". This is likely because, in long-horizon tasks, planners will output suboptimal actions with a higher probability. On the other hand, the adaptation module can assist the planner in better sensing the environment, leading to a general improvement across nearly all areas.

Task Type	Avg. Sub-Goal Len.
Examine	2.07
Pick	2.48
Pick Two	5.70
Stack	5.63
Clean	7.25
Cool	10.36
Heat	12.78

6 Conclusion

This paper explores an effective few-shot framework for Embodied Instruction Following. We approach the task as a POMDP and design a closed-loop Hindsight Planner equipped with an adaptation module to enhance the agent's environmental sensing capabilities. Compared to previous open-loop, supervised methods, our approach is more robust and performs better. Furthermore, the planner incorporates a novel hindsight method that enables it to learn from suboptimal trajectories. we hope our work inspires future research in this area.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022. URL https://arxiv.org/abs/2204.01691.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018. URL https://arxiv.org/abs/1707.01495.

Suvaansh Bhambri, Byeonghwi Kim, and Jonghyun Choi. Multi-level compositional reasoning for interactive instruction following, 2024. URL https://arxiv.org/abs/2308.09387.

Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. A persistent spatial semantic representation for high-level natural language instruction execution. *Cornell University - arXiv,Cornell University - arXiv*, Jul 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

David Chapman. Planning for conjunctive goals. *Artif. Intell.*, 32:333–377, 1987. URL https://api.semanticscholar.org/CorpusID:1525549.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers, 2023. URL https://arxiv.org/abs/2212.10559.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning, 2024. URL https://arxiv.org/abs/2301.00234.

Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. Rewriting history with inverse rl: Hindsight inference for policy improvement, 2020. URL https://arxiv.org/abs/2002.11089.

Hiroki Furuta, Yutaka Matsuo, and Shixiang Shane Gu. Generalized decision transformer for offline hindsight information matching, 2022. URL https://arxiv.org/abs/2111.10364.

Dibya Ghosh, Abhishek Gupta, Justin Fu, Ashwin Reddy, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals without reinforcement learning. *ArXiv*, abs/1912.06088, 2019.

Jiaming Guo, Rui Zhang, Xishan Zhang, Shaohui Peng, Qi Yi, Zidong Du, Xing Hu, Qi Guo, and Yunji Chen. Hindsight value function for variance reduction in stochastic dynamic environment, 2021. URL https://arxiv.org/abs/2107.12216.

Elad Hazan, Sham M. Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration, 2019. URL https://arxiv.org/abs/1812.02690.

Byeonghwi Kim, Jinyeon Kim, Yuyeong Kim, Cheolhong Min, and Jonghyun Choi. Context-aware planning and environment-aware memory for instruction following embodied agents, 2024. URL https://arxiv.org/abs/2308.07241.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots, 2021. URL https://arxiv.org/abs/2107.04034.

Jonathan N. Lee, Alekh Agarwal, Christoph Dann, and Tong Zhang. Learning in pomdps is sample-efficient with hindsight observability, 2023. URL https://arxiv.org/abs/2301.13857.

Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching, 2020. URL https://arxiv.org/abs/1906.05274.

Alexander C. Li, Lerrel Pinto, and Pieter Abbeel. Generalized hindsight for reinforcement learning, 2020. URL https://arxiv.org/abs/2002.11708.

Zhihan Liu, Hao Hu, Shenao Zhang, Hongyi Guo, Shuqi Ke, Boyi Liu, and Zhaoran Wang. Reason for future, act for now: A principled framework for autonomous llm agents with provable sample efficiency, 2024. URL https://arxiv.org/abs/2309.17382.

SoYeon Min, DevendraSingh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. Film: Following instructions in language with modular methods. *Learning,Learning*, Oct 2021.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals, 2020. URL https://arxiv.org/abs/2004.00784.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control, 2020. URL https://arxiv.org/abs/1802.09081.

Suyeon Shin, Sujin jeon, Junghyun Kim, Gi-Cheon Kang, and Byoung-Tak Zhang. Socratic planner: Inquiry-based zero-shot planning for embodied instruction following, 2024. URL https://arxiv.org/abs/2404.15190.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. doi: 10.1109/cvpr42600.2020.01075. URL http://dx.doi.org/10.1109/cvpr42600.2020.01075.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B. Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. Generalized planning in pddl domains with pretrained large language models, 2023. URL https://arxiv.org/abs/2305.11014.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models, 2023. URL https://arxiv.org/abs/2212.04088.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference, 2022. URL https://arxiv.org/abs/2111.02080.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning in language models, 2024. URL https://arxiv.org/abs/2305.16582.

Yichi Zhang and Joyce Chai. Hierarchical task learning from language instructions with unified transformers and self-monitoring, 2021. URL https://arxiv.org/abs/2106.03427.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models, 2023. URL https://arxiv.org/abs/2309.01219.

Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies, 2019. Publisher Copyright: © 7th International Conference on Learning Representations, ICLR 2019. All Rights Reserved.; 7th International Conference on Learning Representations, ICLR 2019 ; Conference date: 06-05-2019 Through 09-05-2019.

A MORE ALGORITHM

In algorithm 2, we present a beam search example of a hindsight planner. During the collection phase, one `Actor` prompted from the ground truth sample pool is required to output W actions for each state, and `Critic` is used to retain the best B actions for the next round of planning. When the search depth U is reached, the best rollout is selected, and the first action from it is returned. At the deployment phase, two `Actors` are prompted with hindsight prompts and ground truth samples. Each `Actor` is required to generate $\frac{W}{2}$ actions.

algorithm 3 outlines the algorithm for the collection phase. To preserve the few-shot assumption, the planner collects suboptimal trajectories from $\mathcal{D}_{\text{gt}}$. During the execution of the current task, this task is specifically excluded from being used as an ICL sample to the planner. We employ a prompt generator ϕ to relabel tasks and mitigate ambiguity in the instructions.

Algorithm 2 LLM Planner: A Beam Search example

Input `Actors`, `Critic`, the initial state s,a generator ψ, search Breadth B, proposal width W and search Depth U
State $S_0 \leftarrow \{s\}$
Action array $A_0 \leftarrow \varnothing$
Get numbers of `Actors` $n \leftarrow len(\texttt{Actors})$
for $u = 0, \ldots, U$ **do**
  for $\texttt{Actor}_\texttt{i}$ in `Actors` **do**
    For each s_u in S_u, invoke $\texttt{Actor}_\texttt{i}$ to propose $\frac{W}{n}$ candidate actions.
  end for
  For each $a_u^{(w)}$ invoke ψ to generate next state $s_{u+1}^{(w)}$
  For each tuple $(s_u, a_u^{(w)}, s_{u+1}^{(w)})$, invoke `Critic` to evaluate the expected cumulative reward $V_{u+1}^{(w)}$
  select B best $(s_u, a_u^{(w)}, s_{u+1}^{(w)})$ with highest V and put them into $S_u \times A_u \times S_{u+1}$
end for
For B preserved rollouts in $S_0 \times A_0 \times \ldots \times S_{U+1}$, invoke `Critic` to evaluate the expected cumulative reward $V_{u+1}^{(b)}$
Select the best rollout $(s_0^*, a_0^*, \ldots, s_{U+1}^*)$
return a_0^*

Algorithm 3 Hindsight Prompt

input: A ground truth sample pool $\mathcal{D}_{gt}$, a prompt generator ϕ.
Initialize initiate `Agent` from $\mathcal{D}_{\text{gt}}$, set $\mathcal{D}_{\text{hind}} \leftarrow \varnothing$.
for sample s in $\mathcal{D}_{\text{gt}}$ **do**
  Extract ground truth rollout R, task description I, PDDL arguments P from s.
  Initialize environment E with s.
  Collect suboptimal trajectories $traj \leftarrow \texttt{Agent}(I, E, D_{\text{gt}}/\{s\})$ (e.g. algorithm 2 of appendix A).
  Rename task description $\widetilde{I} \leftarrow \phi(P)$.
  Get reflection Think $\leftarrow \texttt{LLM}(\widetilde{I}, \text{traj}, R)$.
  Relabel trajectory $\text{prompt}_{\text{actor}} \leftarrow \texttt{LLM}(\widetilde{I}, \text{traj}, R, \text{Think})$.
  Generate critic from suboptimal trajectory $\text{prompt}_{\text{critic}} \leftarrow \texttt{LLM}(\widetilde{I}, \text{traj}, R)$.
  Append $\text{prompt}_{\text{actor}}$, $\text{prompt}_{\text{critic}}$ to $\mathcal{D}_{\text{hind}}$.
end for
Build hindsight sample pool $\mathcal{D} = \mathcal{D}_{\text{gt}} \bigcup \mathcal{D}_{\text{hind}}$.
Initial $\texttt{Actor}_{\texttt{gt}}$, `Adapter` from $\mathcal{D}_{gt}$, initial `Critic`, $\texttt{Actor}_{\texttt{hind}}$ from $\mathcal{D}_{\text{hind}}$.
Return $\texttt{Actor}_\theta$, `Critic`, `Adapter` for any $\theta \in \{\texttt{gt}, \texttt{hind}\}$.

B PROMPTS

B.1 PROMPTS FOR PLANNER

Here, we display prompts for various components here. The <base_info> defines the role descriptions while the <samples> provide in-context examples for `Actors`, `Critic` and `Adapter`.

We first show the role description for `Actors`, `Critic` and `Adapter`.

```
<base_info> of Actor
Interact with a household to solve a task.
At each step, you will be provided with the previous observations and
action pairs.

Important: You **are required** to return an action.

The answer should contain two parts, the action type and a target.

The allowed types of actions are:

OpenObject, CloseObject, PickupObject, PutObject, ToggleObjectOn,
ToggleObjectOff, SliceObject, Stop

The target of OpenObject, CloseObject, PickupObject, ToggleObjectOn,
ToggleObjectOff, SliceObject is the object agent interacts with, and
the target of PutObjectis the place to put the object.

Stop should end with NIL.Note if all requirements are satisfied, you
just need to output Stop
```

```
<base_info> of Critic
You are a value critic of states in a household task. You would be
given a task description, some observations and actions, you need to
give a critic about them. **Note Your critic should end with format:
the value is a/b=...**

The allowed types of actions are: OpenObject,CloseObject,PickupObject,
PutObject,ToggleObjectOn,ToggleObjectOff,SliceObject,Explore,Stop

The target of OpenObject,CloseObject,PickupObject,ToggleObjectOn,Toggl
eObjectOff,SliceObject is the object agent interacts with and the
target of PutObjectis the place to put the object.

Explore and Stop should be followed with NIL.Note if all requirements
are satisfied, you just need to output Stop. You might need to
OpenObject so you can see the object you need to interact with.
```

```
<base_info> of Adapter
Predict the necessary components for the following household task:
-**Moveable Receptacle (mrecep_target)**: Identify any container or
vessel required for the task. Return `None` if not applicable.
-**Object Slicing (object_sliced)**: Determine if the object needs to
be sliced. Provide a boolean value (`True` for yes, `False` for no).
-**Object Target (object_target)**: Identify the specific object that
is the focus of the task and will be interacted with. This could be the
item that needs to be moved, cleaned, heated, cooled, sliced or
examined.
-**Parent Target (parent_target)**: Specify the final resting place for
the object or its parts. Return `None` if there is no designated
location.
-**Toggle Target (toggle_target)**: Indicate any appliance or device
that must be toggled during the task. Return `None` if no toggling is
required.
-**Object State (object_state)**: Indicate whether the target object
needs to be clean, heat, or cool. Return 'None' if no such action is
required.
```

```
-**Two Objects (two_object)**: Specify whether the task requires the
agent to handle and place two *identical* objects into the parent
target location. Set to True if needed, otherwise False.  Note that
this parameter should be True only when the task demands picking and
placing two of the *same* items.
-**Note that the objects you need to predict might not been seen yet.
```

We then present the `<samples>` to the `Actors`, the `Critic`, and the `Adapter`. Since there are 140 samples for each component, we select only 2 samples from each to demonstrate.

<samples> for Adapter

```
Task: Place a cup in the coffee maker.
The objects you seen are: Bread,ButterKnife,Cabinet,Chair,CoffeeMachin
e,CounterTop,Cup,DishSponge,Drawer,Fork,Fridge,GarbageCan,Lettuce,Micr
owave,Mirror,Mug,Pan,Plate,Pot,SaltShaker,Sink,SoapBottle,Spatula,Spoo
n,StoveBurner,StoveKnob,DiningTable,SideTable,Toaster,Window
Predict: mrecep_target: None
object_sliced: False
object_target: Mug
parent_target: CoffeeMachine
toggle_target: None
object_state: cool
two_object: False
Task: Warm a cup to make coffee
The objects you seen are: Apple,Bread,ButterKnife,Cabinet,CoffeeMachin
e,CounterTop,Cup,Drawer,Egg,Fork,Fridge,GarbageCan,HousePlant,Kettle,K
nife,Ladle,Lettuce,Microwave,Mirror,Pan,PepperShaker,Pot,Potato,SaltSh
aker,Sink,Spatula,StoveBurner,StoveKnob,Toaster,Tomato,Window
Predict: mrecep_target: None
object_sliced: False
object_target: Mug
parent_target: CoffeeMachine
toggle_target: None
object_state: heat
two_object: False
```

<samples> for Actor$_{gt}$

```
Task:Place a cup in the coffee maker.
The objects you have seen are:Bread,ButterKnife,Cabinet,Chair,CoffeeMa
chine,CounterTop,Cup,DishSponge,Drawer,Fork,Fridge,GarbageCan,Lettuce,
Microwave,Mirror,Mug,Pan,Plate,Pot,SaltShaker,Sink,SoapBottle,Spatula,
Spoon,StoveBurner,StoveKnob,DiningTable,SideTable,Toaster,Window
Act: OpenObject : Cabinet
>OK
Act: PickupObject : Mug
>OK
Act: CloseObject : Cabinet
>OK
Act: OpenObject : Fridge
>OK
Act: PutObject : Fridge
>OK
Act: CloseObject : Fridge
>OK
Act: OpenObject : Fridge
>OK
Act: PickupObject : Mug
>OK
Act: CloseObject : Fridge
>OK
```

```
Act: PutObject : CoffeeMachine
>OK
Act: Stop : NIL
>OK

Task:Pick up tissues, drop them in the tub.
The objects you have seen are:Bathtub,Cabinet,Candle,Cloth,CounterTop,
Mirror,Plunger,ScrubBrush,Sink,SoapBar,SoapBottle,Toilet,ToiletPaper,T
oiletPaperHanger,Towel,TowelHolder,Window
Act: PickupObject : Cloth
>OK
Act: PutObject : Bathtub
>OK
Act: Stop : NIL
>OK
```

<samples> for $\text{Actor}_{\text{hind}}$

```
Task:Place a cup in the coffee maker.
The objects you have seen are:Bowl, Bread, ButterKnife, Cabinet, Chair,
CoffeeMachine, CounterTop, Cup, DishSponge, Drawer, Fridge, GarbageCan,
Lettuce, LightSwitch, Microwave, Mirror, Mug, Pan, PepperShaker, Plate,
SaltShaker, Sink, SoapBottle, Spatula, Spoon, StoveBurner, StoveKnob,
DiningTable, SideTable, Toaster, Window
Act: OpenObject : Cabinet
>OK
Act: PickupObject : Cup
>OK
Act: CloseObject : Cabinet
>OK
Act: PutObject : DiningTable
>OK
Act: OpenObject : Fridge
>OK
Act: PickupObject:Cup
>OK
Act: PutObject:Fridge
>OK
Act: CloseObject:Fridge
>OK
Act: OpenObject:Fridge
>OK
Act: PickupObject:Cup
>OK
Act: CloseObject:Fridge
>OK
Act: PutObject:CoffeeMachine
>OK
Act: Stop : NIL
>OK

Task:Pick up tissues, drop them in the tub.
The objects you have seen are:Bathtub, Cabinet, Candle, Cloth,
CounterTop, GarbageCan, HandTowel, HandTowelHolder, LightSwitch,
Mirror, Painting, ScrubBrush, Shelf, ShowerDoor, ShowerGlass, Sink,
SoapBottle, Television, Toilet, ToiletPaperHanger, Towel, TowelHolder,
Window
Act: PickupObject : TissueBox
>OK
Act: PutObject:CounterTop
>OK
Act: PickupObject:Cloth
>OK
Act: PutObject:Bathtub
```

```
>OK
Act: Stop : NIL
>OK
```

Having demonstrated the <base_info> and <samples>, we can now present the prompt template for the Actors, Critic, and Adapter. Note that the prompt of Actor_{gt} and $\text{Actor}_{\text{hind}}$ differs in <samples>. The <object_list> indicates the objects the agent has seen in the environment. Meanwhile, the <PDDL_predicted> refers to the output of the Adapter, and the <K> indicates the number of samples in each component. To facilitate better comprehension by LLMs, we convert the PDDL arguments into a natural language description. The <previous_history> includes the previous actions executed by the agent, enabling the planner to make better decisions based on this information. Concurrently, we employ a prompt generator that reviews the <previous_history> and outputs <history_information> exclusively to assist LLMs in identifying the objects being held and the open/closed status of containers.

Prompt of Adapter

```
<Adapter_base_info>
Here are <K> examples:
<Adapter_samples>
Your task is: <task_inst>
The objects you have seen are: <object_list>
```

Prompt of Critic

```
<Critic_base_info>
Here are <K> examples:
<Critic_samples>
Your task is: <task_inst>
Your knowledge about this task is: <PDDL_predicted>
The objects you have seen are: <object_list>
previous_history
Based on the **actions** and **Your knowledge about this task** , write
a Critic.
Critic:
```

Prompt of Actor

```
<Actor_base_info>
Here are <K> examples:
<Actor_samples>
Your task is: <task_inst>
Your knowledge about this task is: <PDDL_predicted>
The objects you have seen are: <object_list>
Your knowledge about the current state is: <history_information>
<previous_history>
Act:
```

B.2 PROMPTS FOR HINDSIGHT

We now present the prompts used to query LLMs in our hindsight method. During the relabeling process for Actor, we first prompt LLMs to generate a <Think> for the suboptimal trajectory, and then we query the LLMs to complete the task based on it. For the relabeling process of the Critic, we directly prompt the LLMs to generate a critic for the suboptimal trajectory. We first present the hindsight samples.

<samples> for Actor Think

```
Task: Put a fork on a table.
groundtruth rollout:
PickupObject:Fork
PutObject:Sink
ToggleObjectOn:Faucet
ToggleObjectOff:Fauce
PickupObject:Fork
PutObject:SideTable
Stop:NIL
the incomplete rollout:
PickupObject:Fork
PutObject:SideTable

Think: According to the groundtruth rollout, in this incomplete
rollout, I don't clean the fork and the fork is on the sidetable, I
need to pick up the fork and use faucet to clean the fork and put it
onto the sidetable.
Task: Put a warmed apple in the fridge.
groundtruth rollout:
PickupObject:Apple
OpenObject:Microwave
PutObject:Microwave
CloseObject:Microwave
ToggleObjectOn:Microwave
ToggleObjectOff:Microwave
OpenObject:Microwave
PickupObject:Apple
CloseObject:Microwave
OpenObject:Fridge
PutObject:Fridge
CloseObject:Fridge
Stop:NIL
the incomplete rollout:
PickupObject : Apple
OpenObject : Fridge
PutObject : Fridge
CloseObject : Fridge
OpenObject : Fridge
PickupObject : Apple
CloseObject : Fridge
OpenObject : Microwave
Think: According to the groundtruth rollout, in this incomplete
rollout, I don't heat the apple and the apple is in the fridge, I need
to open the fridge, pickup the apple and use microwave to heat the
apple, then I should put the apple back into the fridge.
```

<samples> for Actor Complete

```
task: Put a fork on a table.
groundtruth rollout:
PickupObject:Fork
PutObject:Sink
ToggleObjectOn:Faucet
ToggleObjectOff:Fauce
PickupObject:Fork
PutObject:SideTable
Stop:NIL
the incomplete rollout:
PickupObject:Fork
PutObject:SideTable
```

```
Think: According to the groundtruth rollout, in this incomplete
rollout, I don't clean the fork and the fork is on the sidetable, I
need to pick up the fork and use faucet to clean the fork and put it
onto the sidetable.
Based on the Think and groundtruth rollout, the new actions append to
the incomplete rollout are:
PickupObject : Fork
PutObject :Sink
ToggleObjectOn : Faucet
ToggleObjectOff : Faucet
PickupObject:  Fork
PutObject:SideTable
Stop : NIL

task: Put a warmed apple in the fridge.
groundtruth rollout:
PickupObject : Apple
OpenObject : Microwave
PutObject : Microwave
CloseObject : Microwave
ToggleObjectOn : Microwave
ToggleObjectOff : Microwave
OpenObject : Microwave
PickupObject : Apple
CloseObject : Microwave
OpenObject : Fridge
PutObject : Fridge
CloseObject : Fridge
Stop:NIL
the incomplete rollout:
PickupObject : Apple
OpenObject : Fridge
PutObject : Fridge
CloseObject : Fridge
OpenObject : Fridge
PickupObject : Apple
CloseObject : Fridge
OpenObject : Microwave
Think: According to the groundtruth rollout, in this incomplete
rollout, I don't heat the apple and the apple is in the fridge, I need
to open the fridge, pickup the apple and use microwave to heat the
apple, then I should put the apple back into the fridge.
Based on the Think and groundtruth rollout, the new actions append to
the incomplete rollout are:
OpenObject : Fridge
PickupObject : Apple
CloseObject : Fridge
PutObject: Microwave
CloseObject: Microwave
ToggleObjectOn: Microwave
ToggleObjectOff : Microwave
OpenObject : Microwave
PickupObject : Apple
CloseObject : Microwave
OpenObject : Fridge
PutObject : Fridge
CloseObject : Fridge
```

<samples> for critic generation

```
Your task is: Put the cooked tomato on the round table
The rollout by agent is: OpenObject : Fridge
PickupObject : Tomato
CloseObject : Fridge
```

```
OpenObject : Microwave
The **ground truth rollout** is:
PickupObject:Tomato OpenObject:Microwave
PutObject:Microwave
CloseObject:Microwave
ToggleObjectOn:Microwave
ToggleObjectOff:Microwave
OpenObject:Microwave
PickupObject:Tomato
CloseObject:Microwave
PutObject:DiningTable
Stop:NIL
Based on the **ground truth rollout** , write a critic
Critic:In this task, I need to do the following things in order: Pick
the tomato and put it into microwave, use microwave to heat it,pick the
tomato from microwave and put it onto the DiningTable.There are 5
subgoals in orde, I only achieved first of them, the value is 1/5=0.2.

Your task is: Put a chilled mug in the bottom cabinet closest to the
fridge.
The rollout by agent is:
PickupObject : Mug
OpenObject : Fridge
PutObject : Fridge
CloseObject : Fridge
OpenObject : Fridge
PickupObject : Mug
CloseObject : Fridge
PutObject : Cabinet
The **ground truth rollout** is:
PickupObject:Mug
OpenObject:Fridge
PutObject:Fridge
CloseObject:Fridge
OpenObject:Fridge
PickupObject:Mug
CloseObject:Fridge
OpenObject:Cabinet
PutObject:Cabinet
CloseObject:Cabinet
Stop:NIL
Based on the **ground truth rollout** , write a critic
Critic:In this task, I need to do the following things in order: pick
the mug and put it into the fridge, pick the mug from the fridge and
put the mug into the cabinet. There are 3 subgoals in all, I achieved 2
of them, this is because I don't open the cabinet, so I can't put the
mug into it, the value is 2/3=0.66
```

We provide the prompts used for querying Actors and Critic, respectively. The <relabeled_task> indicates the task rewritten based on its PDDL, as detailed in section 5.1. The <gt_rollout> represents the ground truth rollout, while the <suboptimal_rollout> denotes the rollout collected by our agent.

prompt of Actor_Think

```
You are a housework agent, you will be given a task, a ground truth
rollout to complete this task, and an incomplete rollout.
Your goal is to consider what action you need to append to the
incomplete rollout to complete the task.
Important: You should use your knowledge to judge what actions need to
do based on the ground truth rollout and incomplete rollout. eg: if the
agent forget to open the fridge, then the action of put object into
fridge should be counted as failed, so you should open the fridge and
put the object into the fridge.
```

```
Important: the openable object (fridge,mrcrowave...) are initially
closed, so you need to open them before put object in it.
Important: You can hold one object in your hand at once.

The allowed types of actions are: OpenObject,CloseObject,PickupObject,PutObject,ToggleObjectOn,ToggleObjectOff,SliceObject,Stop
The target of actions like OpenObject, CloseObject, PickupObject,
ToggleObjectOn, ToggleObjectOff, and SliceObject is the object the
agent interacts with, whereas the target of PutObject is the location
where the object is to be placed.
The 'Stop' action should be followed by 'NIL'. Note that if all
requirements are met, you only need to output 'Stop'. Remember that you
can only pick up one item at a time, so you must put down the object in
your hand before picking up a new one.

Here are k examples:
<Actor_Think_samples>
Task: <relabeled_task>
Ground truth rollout: <gt_rollout>
The incomplete rollout: <suboptimal_rollout>
Think:
```

prompt of Actor_Complete

```
You are a housework agent, you will be given a task, a ground truth
rollout to complete this task,  an incomplete rollout, and a think
about the incomplete rollout.
Your goal is to finish the incomplete rollout based on the groundtruth
rollout and your think.
Important: You can only output the needed actions,seperated by '
', you must not output other things

The allowed types of actions are: OpenObject,CloseObject,PickupObject,PutObject,ToggleObjectOn,ToggleObjectOff,SliceObject,Stop
The target of actions like OpenObject, CloseObject, PickupObject,
ToggleObjectOn, ToggleObjectOff, and SliceObject is the object the
agent interacts with, whereas the target of PutObject is the location
where the object is to be placed.
The 'Stop' action should be followed by 'NIL'. Note that if all
requirements are met, you only need to output 'Stop'. Remember that you
can only pick up one item at a time, so you must put down the object in
your hand before picking up a new one.

Here is k examples:
<Actor_Complete_samples>
Task: <relabeled_task>
Ground truth rollout: <gt_rollout>
The incomplete  rollout: <suboptimal_rollout>
Think: <Think>
Based on the Think and groundtruth rollout, the new actions append to
the incomplete rollout are:
```

critic generation prompt

```
You will be provided with a household task roll-out conducted by an
agent and a ground truth roll-out. Your task is to write a critic of
the agent's roll-out based on the **ground truth rollout** The critic
should follow the form:In this task, I need do the follwing things in
order:... There are ... subgoals I need to achieve,My current state
achieve ...
Important: You should use your knowledge to judge how many subgoals are
achieved. eg: if the agent forget to open the fridge, then the action
of put object into fridge should not counted.
```

```
Important: Your critic should end with "the value is a/b=.." You can
round it into 2 decimal.
Important: You should write your critic based on given format, you
should't output other things.
Important: You shouldn't mention about ground truth rollout in your
critic.
Here are examples: <Critic_samples>
Your task is: <relabelled_task>
The rollout by agent is:: <suboptimal_rollout>
The **ground truth rollout** is: <gt_rollout>
Based on the **ground truth rollout** , write a critic
Critic:
```