KnowAgent: Knowledge-Augmented Planning for LLM-Based Agents

Yuqi Zhu♣♡, Shuofei Qiao♣♡, Yixin Ou♣♡, Shumin Deng♠, Ningyu Zhang♣♡*, Shiwei Lyu♢, Yue Shen♢, Lei Liang♢, Jinjie Gu♢, Huajun Chen♣♡*

♣ Zhejiang University ♡ ZJU-Ant Group Joint Research Center for Knowledge Graphs
♢ Ant Group ♠ National University of Singapore
{zhuyuqi,zhangningyu}@zju.edu.cn

https://zjunlp.github.io/project/KnowAgent/

Abstract

Large Language Models (LLMs) have demonstrated great potential in complex reasoning tasks, yet they fall short when tackling more sophisticated challenges, especially when interacting with environments through generating executable actions. This inadequacy primarily stems from the lack of built-in action knowledge in language agents, which fails to effectively guide the planning trajectories during task solving and results in *planning hallucination*. To address this issue, we introduce **KnowAgent**, a novel approach designed to enhance the planning capabilities of LLMs by incorporating explicit action knowledge. Specifically, **KnowAgent** employs an *action knowledge base* and a *knowledgeable self-learning* strategy to constrain the action path during planning, enabling more reasonable trajectory synthesis, and thereby enhancing the planning performance of language agents. Experimental results on HotpotQA and ALFWorld based on various backbone models demonstrate that **KnowAgent** can achieve comparable or superior performance to existing baselines. Further analysis indicates the effectiveness of **KnowAgent** in terms of *planning hallucinations* mitigation[1].

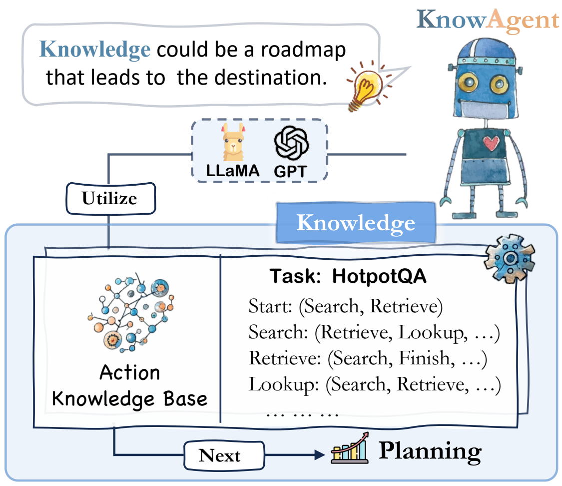

Figure 1: **The overview of KnowAgent**. An agent could leverage external action knowledge base to address and solve complex planning challenges.

1 Introduction

As artificial intelligence (AI) advances, language agents are becoming increasingly vital for solving complex problems (Zhang et al., 2023; Sumers et al., 2024; Yang et al., 2024). These agents, built around Large Language Models (LLMs), enhance their task planning capabilities through a variety of strategies including task decomposition (Wei et al., 2022; Yao et al., 2023a; Wang et al., 2023a; Team, 2023), reflection (Shinn et al., 2023; Sun et al., 2023a), collaborative division of labor (Hong et al., 2023; Chen et al., 2023c; Yin et al., 2023; Qiao et al., 2024), and the utilization of external tools (Schick et al., 2023; Qiao et al., 2023a). Despite the effectiveness of current prompting techniques in providing good planning abilities for some closed-source language models, these methods are often limited by the model's intrinsic understanding capabilities and the scope of knowledge it was trained on. To meet the demands for broad application and customization in different areas such as question-answering (Yao et al., 2023c; Yin et al., 2023), web browsing (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2023a), robotics (Ichter et al., 2022; Ding et al., 2023) and so on, researchers are exploring *Agent Tuning* as a means to augment model capabilities (Chen et al., 2023a; Zeng et al., 2023; Pan et al., 2023). This involves fine-tuning models through the synthesis of task-specific trajectories, enabling them to undertake a series of effective actions to complete tasks, thereby enhancing their ability to handle complex situations.

However, when it comes to executing planning tasks, especially in open-source models, there re-

*Corresponding author.
[1]Code is at https://github.com/zjunlp/KnowAgent.

main issues (Liu et al., 2023a; Valmeekam et al., 2023; Guan et al., 2023). Frequently, models generate plans that violate established knowledge rules or commonsense (Ding et al., 2023), a phenomenon we name as *planning hallucination*. This term describes scenarios where models might generate unnecessary or conflicting action sequences, such as "`attempting to look up information without performing a search operation`" or "`trying to pick an apple from a table without verifying the presence of both the table and the apple`".

To address these issues, we propose **KnowAgent** that focuses on leveraging external action knowledge to enhance synthetic trajectories with the goal of resolving *planning hallucination* (see Figure 1). Our development is grounded on several key steps: Initially, we create an extensive *action knowledge base*, which amalgamates action planning knowledge pertinent to specific tasks. This database acts as an external reservoir of information, steering the model's action generation process. Subsequently, by converting action knowledge into text, we enable the model to deeply understand and utilize this knowledge in creating action trajectories. Finally, through a *knowledgeable self-learning* phase, we use trajectories developed from the model's iterative processes to continually improve its understanding and application of action knowledge. This process not only strengthens the agents' planning abilities but also enhances their potential for application in complex situations.

Experimental results on HotpotQA (Yang et al., 2018) and ALFWorld (Shridhar et al., 2021) based on various backbone models demonstrate that **KnowAgent** can achieve comparable or superior performance to existing baselines. Further analysis indicates the effectiveness of **KnowAgent** in terms of *planning hallucinations* mitigation. We summarize our contributions as follows:

- We introduce KnowAgent that employs *knowledgeable self-learning* to incorporate external *action knowledge* into models. This advancement presents an innovative method for incorporating external knowledge to refine and augment the intrinsic planning abilities of language agents.

- We conduct comprehensive experiments that demonstrate KnowAgent can match or surpass other benchmark models on the HotpotQA and ALFWorld datasets.

- Further analysis validates the effectiveness of incorporating *action knowledge* for planning purposes. We also showcase the possibility of employing manually refined action knowledge from LLMs, thereby reducing human labor and enhancing performance.

2 Background

Language agents observe the external world primarily by generating inner thoughts and executable actions. In this paper, we follow and further enhance the planning trajectory format proposed in Yao et al. (2023b) to train and evaluate our KnowAgent. Traditionally, a planning trajectory τ can be represented by a triplet of Thought-Action-Observation $(\mathcal{T}, \mathcal{A}, \mathcal{O})$, where $\mathcal{T}$ indicates the inner thoughts of the language agent, $\mathcal{A}$ signifies executable actions, and $\mathcal{O}$ represents the feedback information from the environment. In terms of this, the trajectory history $\mathcal{H}$ at time t can be defined as follows:

$$\mathcal{H}_t = (\mathcal{T}_0, \mathcal{A}_0, \mathcal{O}_0, \mathcal{T}_1, ..., \mathcal{T}_{t-1}, \mathcal{A}_{t-1}, \mathcal{O}_{t-1}) \quad (1)$$

Then, the language agent is reinforced to generate $\mathcal{T}_t$ and $\mathcal{A}_t$ based on the history. Given a parameterized probabilistic language agent π with parameters θ, the process of generating the next step's thought based on $\mathcal{H}_t$ can be represented as:

$$p(\mathcal{T}_t|\mathcal{H}_t) = \prod_{i=1}^{|\mathcal{T}_t|} \pi_\theta(\mathcal{T}_t^i|\mathcal{H}_t, \mathcal{T}_t^{<i}), \quad (2)$$

where $\mathcal{T}_t^i$ and $|\mathcal{T}_t|$ are the i-th token and the length of $\mathcal{T}_t$ respectively. Subsequently, the action $\mathcal{A}_t$ will be determined based on $\mathcal{T}_t$ and $\mathcal{H}_t$:

$$p(\mathcal{A}_t|\mathcal{H}_t, \mathcal{T}_t) = \prod_{j=1}^{|\mathcal{A}_t|} \pi_\theta(\mathcal{A}_t^j|\mathcal{H}_t, \mathcal{T}_t, \mathcal{A}_t^{<j}). \quad (3)$$

Similarly, $\mathcal{A}_t^j$ and $|\mathcal{A}_t|$ denote the j-th token and the length of $\mathcal{A}_t$ respectively. Lastly, the feedback result of the action $\mathcal{A}_t$ will be treated as the observation $\mathcal{O}_t$ and added to the trajectory, generating a new round of trajectory $\mathcal{H}_{t+1}$. It's important to note that $\mathcal{A}_i$ here specifically means the actions in the trajectory, which **is identical to** the action a_i in the discussion of the action set E_a later on.

3 KnowAgent

In this section, we offer a detailed introduction to KnowAgent (See Figure 2), focusing on three

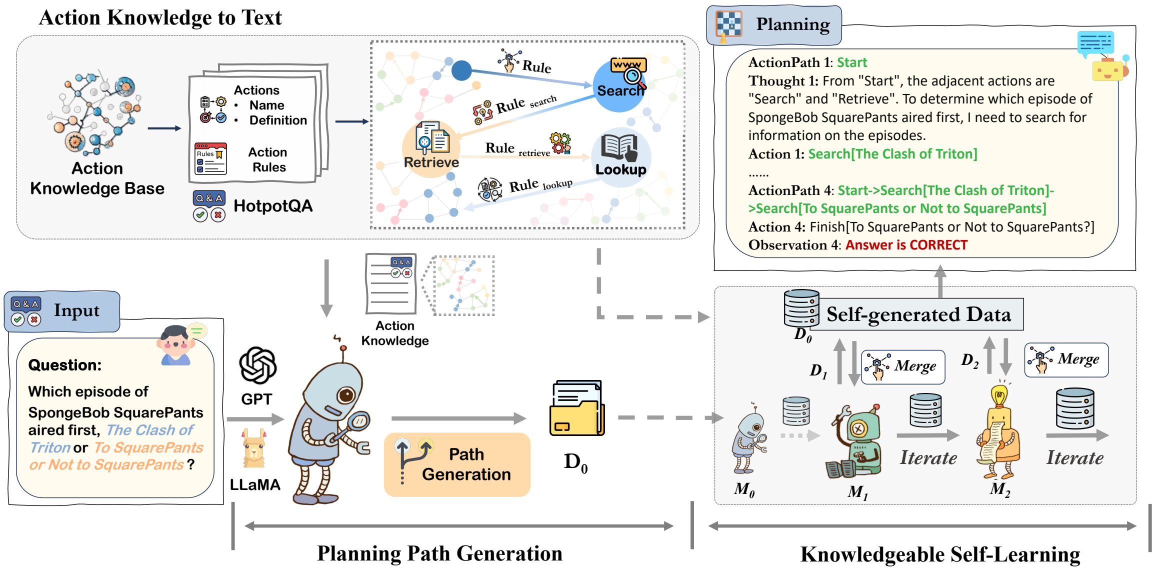

Figure 2: **The overall framework of KNOWAGENT.** Initially, *Action Knowledge to Text* converts task-specific action knowledge into textual descriptions. Next, *Planning Path Generation* uses prompts and this knowledge to lead LLMs in planning path creation. Lastly, in *Knowledgeable Self-Learning*, the model iteratively optimizes using generated planning trajectories to improve performance.

core aspects. First, we define *action knowledge* in §3.1. Next, we describe how *action knowledge* is utilized to generate planning paths (§3.2). Lastly, we detail the refinement of the paths through a *knowledgeable self-learning* mechanism. (§3.3).

3.1 Definition of Action Knowledge

Action. $E_a = \{a_1, ..., a_{N-1}\}$ signifies a set of actions, which encompasses the discrete action that LLMs must undertake to accomplish specific tasks.

Action Rules. $\mathcal{R} = \{r_1, ..., r_{N-1}\}$ outlines the rules that determine the logic and sequence of action transitions within the model. These rules directly dictate permissible action transitions $r_k : a_i \rightarrow a_j$, based on the inherent relationships among actions or task-specific requirements.

Action Knowledge. Action Knowledge, represented as $(E_a, \mathcal{R})$, comprises a defined set of actions E_a and the rules $\mathcal{R}$ governing their transitions. The combination of action knowledge for different tasks forms an action knowledge base, also known as ***Action KB***. The knowledge base will then serve as essential guidance for generating actions and formulating decisions, essential for reducing potential *plan hallucination* problems.

Strategies for Extracting Action Knowledge. Given the diverse action knowledge involved in various tasks, fully manual construction is both time-consuming and labor-intensive. To address this challenge, considering the strong performance of LLMs on such tasks (Liu et al., 2023a; Ouyang and Li, 2023), we utilize GPT-4 (OpenAI, 2023) for initial construction, followed by manual refinement. In § 4.3, we provide a detailed comparison of the effectiveness of these two approaches.

3.2 Planning Path Generation with Action Knowledge

3.2.1 Action Knowledge to Text

Figure 3 illustrates the conversion process from action knowledge to text. Initially, we establish the action knowledge base by identifying actions pertinent to the task's specific needs, utilizing previous dataset analyses and the inherent knowledge of LLMs. This information is then converted into text format to facilitate subsequent operations. As an illustration, we reference one action rule in HotpotQA (Yang et al., 2018) - `Search: (Search, Retrieve, Lookup, Finish)`. This rule signifies that from `Search`, several pathways are viable: an action may continue as `Search`, evolve into `Retrieve` or `Lookup`, or advance towards `Finish`.

3.2.2 Path Generation

Harnessing action knowledge, the model utilizes this insight to streamline the task's planning process. It achieves this by formulating a coherent planning path, guided by the application of action

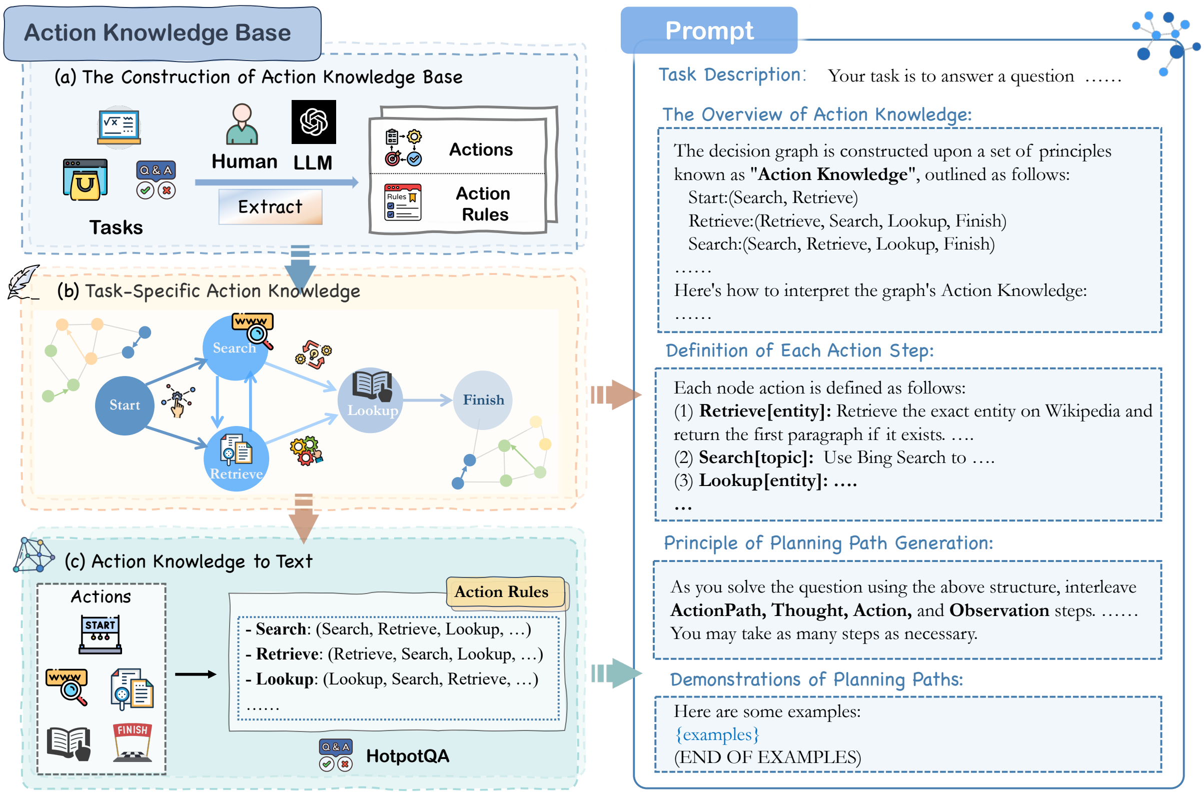

Figure 3: The Path Generation process of KnowAgent.

rules $\mathcal{R}_1 \wedge \mathcal{R}_2 \wedge \ldots \Rightarrow \mathcal{P}$. To facilitate path generation, we develop specialized prompts that extend beyond basic **Task Description**, integrating segments as illustrated in Figure 3.

Our approach is thoroughly grounded in action knowledge and unfolds across four key segments: (1) It starts with an **Overview of Action Knowledge** to set the foundational concepts and rules. (2) This is followed by the **Definition of Each Action Step**, detailing the operational aspects and significance of each action. (3) Following this, the **Principle of Planning Path Generation** delves into the constraints on output generation. (4) And finally, **Demonstrations of Planning Paths** provide practical examples, acting as a beacon of inspiration for adapting these strategies across various contexts. Each of these segments plays an essential role in expressing action knowledge, specifying actions, and clarifying the process of leveraging action knowledge for planning path generation. **It's essential to understand the distinction between path and trajectory in this context**. The **path** exclusively represents the series of actions undertaken by the agent, while the **trajectory** includes the model's complete output during the problem-solving process, incorporating the path as part of its structure.

Here we briefly outline the process of **trajectory** synthesis. This trajectory, denoted as τ, is composed of many planned quadruples. Each quadruple $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{O})$, encapsulates the action path $\mathcal{P}$, the agent's internal thoughts processes $\mathcal{T}$, executable actions $\mathcal{A}$, and environmental feedback $\mathcal{O}$. The historical trajectory is reformulated as:

$$\begin{aligned} \mathcal{H}_t = (&\mathcal{P}_0, \mathcal{T}_0, \mathcal{A}_0, \mathcal{O}_0, \ldots, \\ &\mathcal{P}_{i-1}, \mathcal{T}_{t-1}, \mathcal{A}_{t-1}, \mathcal{O}_{t-1}) \end{aligned} \tag{4}$$

Based on this historical trajectory, the agent is poised to generate a new action path, thought process, and action. Considering a parameterized probabilistic language agent π with parameters θ, the mechanism for generating the subsequent action path, contingent on $\mathcal{P}_t$, is expressed as:

$$p(\mathcal{P}_t|\mathcal{H}_t) = \prod_{k=1}^{|\mathcal{P}_t|} \pi_\theta(\mathcal{P}_t^k|\mathcal{H}_t, \mathcal{P}_t^{<k}), \tag{5}$$

Here $\mathcal{P}_t^k$ and $|\mathcal{P}_t|$ represent the k-th token and the total length of $\mathcal{P}_t$. And then we extend the approach used in Equation 2 and 3. The process of deriving thoughts and actions can be reformulated as:

$$p(\mathcal{T}_t|\mathcal{H}_t, \mathcal{P}_t) = \prod_{i=1}^{|\mathcal{T}_t|} \pi_\theta(\mathcal{T}_t^i|\mathcal{H}_t, \mathcal{P}_t, \mathcal{T}_t^{<i}), \quad (6)$$

$$p(\mathcal{A}_t|\mathcal{H}_t, \mathcal{P}_t, \mathcal{T}_t) = \prod_{j=1}^{|\mathcal{A}_t|} \pi_\theta(\mathcal{A}_t^j|\mathcal{H}_t, \mathcal{P}_t, \mathcal{T}_t, \mathcal{A}_t^{<j}). \quad (7)$$

3.3 Planning Path Refinement via Knowledgeable Self-Learning

In this phase, we introduce *knowledgeable self-learning*. Our goal is to help the model understand the action knowledge more deeply through iterative fine-tuning. As shown in Algorithm 1, our approach begins with an initial training set D_0 and an untrained model M_0, leading to the synthesis of initial trajectories $T_0 = \{\tau_1, \tau_2, \ldots, \tau_n\}$. After filtering, these initial outcomes inform further training, producing a preliminary model version, M_1. Subsequently, M_1 undergoes re-evaluation on D_0 to create new trajectories $T_1 = \{\tau_1', \tau_2', \ldots, \tau_n'\}$. These trajectories, alongside T_0, undergo a filtering and merging process based on action knowledge This refined set of trajectories is then utilized to fine-tune the model, resulting in an improved version, M_2. We continue iterating until the performance improvement on M_{test} becomes small, at which point we halt the iteration process.

Knowledge-Based Trajectory Filtering and Merging. Our *knowledgeable self-learning* approach enhances trajectory quality through two key phases: (1) **Filtering**: We start by selecting correct trajectories, T_{correct}, based on their outcomes. Specifically for task HotpotQA, we apply action knowledge to further refine these trajectories. This refinement involves removing any trajectories that do not align with the provided AK_m, particularly those with invalid actions or disordered action sequences. (2) **Merging**: We then merge trajectories generated by models across different iterations. For trajectories addressing the same task, We refine them based on efficiency, specifically retaining the more efficient (shorter path) trajectories ensuring optimal problem-solving effectiveness.

Algorithm 1: Trajectory Synthesis and Knowledgeable Self-Learning

Input: AK_m: Task-specific Action Knowledge from $Action\ KB$, including a set of actions E_a and action rules R; D_0: Initial training set; D_{test}: Testing set.
Output: Optimized models $M = \{M_1, M_2, \ldots\}$.
Initialize: Model M_0.
for $i = 0$ ***until*** *test performance stabilizes* **do**
  if $i = 0$ **then**
    // Synthesize initial trajectories.
    $T_i \leftarrow Traj(M_i, AK_m, D_0)$
    // Fiter trajectories.
    $T_i' \leftarrow Filter(T_i, AK_m)$
    // Initial fine-tuning.
    $M_{i+1} \leftarrow Tune(T_i', M_i)$
  else
    // Synthesize trajectories.
    $T_i \leftarrow Traj(M_i, AK_m, D_0)$
    // Filter and merge trajectories.
    $T_i' \leftarrow FilterAndMerge(T_i, T_{i-1}, AK_m)$
    // Further fine-tuning.
    $M_{i+1} \leftarrow Tune(T_i', M_i)$
  // Performance check
  if $\Delta Perf(M_{i+1}, M_i, D_{test}) \leq \epsilon$ **then**
    break
return *Optimized models M*

4 Experiments

4.1 Settings

We evaluate KnowAgent on HotpotQA (Yang et al., 2018) and ALFWorld (Shridhar et al., 2021). We employ `Llama-2-{7,13,70}b-chat` (Touvron et al., 2023) as the backbone models, and also apply KnowAgent to Vicuna (Zheng et al., 2023) and Mistral (Jiang et al., 2023). We compare KnowAgent with various baselines including CoT (Wei et al., 2022), ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023) and FiReAct (Chen et al., 2023a). More details about the datasets, evaluation metrics, baselines, and training hyper-parameters can be seen in Appendix A.

4.2 Main Results

KnowAgent vs. Prompt-based Methods. In Table 1, we present the F1 scores and success rates for KnowAgent and various prompt-based methods evaluated on HotpotQA and ALFWorld. Across both datasets, KnowAgent on 7b, 13b, and 70b models consistently outperforms the prompt-based baselines. Notably, the 13b model achieves a performance increase of ↑15.09% and ↑37.81% over ReAct on the two dataset. Additionally, disparities in effectiveness among different prompt methods are observed, which aligns with current research efforts that focus on enhancing models' capabilities to handle complex tasks through diverse strategies such as multi-agent specialization. Specifically, our investigation is par-

Backbone	Strategy	Method	HotpotQA				ALFWorld
			Easy	*Medium*	*Hard*	*Average*	
Llama-2 7B-chat	Prompting	CoT (Wei et al., 2022)	35.80	26.69	18.20	26.90	-
	Prompting	ReAcT (Yao et al., 2023b)	25.14	19.87	17.39	20.80	14.18
	Prompting	Reflexion (Shinn et al., 2023)	35.55	28.73	24.35	29.54	6.34
	Fine-tuning	FiReAct (Chen et al., 2023a)	40.56	31.70	24.13	32.13	25.38
	Fine-tuning	KNOWAGENT-7B	**40.80**	**32.49**	**27.12**	**33.47**	**29.35**
Llama-2 13B-chat	Prompting	CoT (Wei et al., 2022)	37.90	25.28	21.64	28.27	-
	Prompting	ReAcT (Yao et al., 2023b)	28.68	22.15	21.69	24.17	20.90
	Prompting	Reflexion (Shinn et al., 2023)	44.43	37.50	28.17	36.70	34.33
	Fine-tuning	FiReAct (Chen et al., 2023a)	**51.95**	33.93	28.88	38.26	50.37
	Fine-tuning	KNOWAGENT-13B	46.97	**37.60**	**33.22**	**39.26**	**58.71**
Llama-2 70B-chat	Prompting	CoT (Wei et al., 2022)	45.37	36.33	32.27	37.99	-
	Prompting	ReAcT (Yao et al., 2023b)	39.70	37.19	33.62	36.83	55.22
	Prompting	Reflexion (Shinn et al., 2023)	48.01	46.35	35.64	43.33	69.40
	Fine-tuning	FiReAct (Chen et al., 2023a)	51.96	47.56	**44.60**	48.04	77.61
	Fine-tuning	KNOWAGENT-70B	**56.75**	**49.90**	37.76	**48.14**	**78.36**

Table 1: **Overall performance of KNOWAGENT on HotpotQA and ALFWorld.** The evaluation metrics are F1 Score (%) and Success Rate (%), respectively. **Strategy** means the agent learning paradigm behind each method. The best results of each backbone are marked in **bold**.

ticularly geared towards leveraging external action knowledge bases to facilitate models in more accurately completing complex tasks. This is achieved by minimizing invalid actions (on HotpotQA) and promoting action sequences that better reflect real-world situations (on ALFWorld), thereby improving model efficiency. Further analysis, especially in relation to invalid actions in HotpotQA, will be discussed later in §4.3.

KNOWAGENT vs. Fine-tuning Methods. Our comparison here focuses on the fine-tuning results of KNOWAGENT versus FiReAct. A significant difference is that FiReAct's fine-tuning data is synthesized by GPT-4, whereas KNOWAGENT used its self-synthesized data. For instance, on HotpotQA, FiReAct employs 500 correct trajectories generated by GPT-4, while KNOWAGENT also uses a 500 training trajectory volume, but only selects the correct quantity of them, approximately 100 to 200 in 13b. The strategy also mirrors in ALFWorld. The outcomes suggest that model-synthesized data, infused with prior knowledge, can achieve results comparable to those produced by more advanced models like GPT-4. Additionally, the study also indicates that iterative fine-tuning enables the model to comprehensively grasp the action knowledge, leading to superior planning performance.

4.3 Analysis

The role of action knowledge grows with the increase of iterations in self-learning. Figure 4 shows the ablation results about action knowledge on HotpotQA with Llama series models. Regardless of the number of iterations, the effect of using action knowledge (*w/ action KB*) is superior to that without action knowledge (*w/o action KB*), indicating that the introduction of action knowledge can effectively enhance the quality of agent planning. Another interesting finding is that as the number of iterations increases, the performance gap between *w/o action KB* and *w/ action KB* becomes more significant, indicating that the advantages of introducing action knowledge become more apparent. We consider that this can be attributed to the virtuous cycle between *action knowledge* and *self-learning*. Under the constraints of action knowledge, the model synthesizes high-quality trajectories for iterative training. In turn, training on more high-quality trajectories allows the model to better learn action knowledge, leading to the generation of even more high-quality trajectories.

Iterative training enhances model proficiency. Figure 5 presents a comparative analysis of the effects of iterative training across different base models. **(1) The number of iterations.** Notably, elevating iterations from 1 to 2 results in a substantial

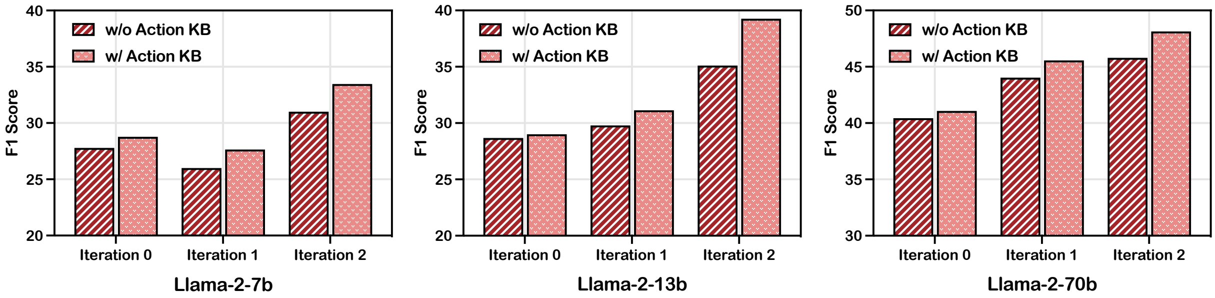

Figure 4: Ablation study on Action Knowledge within Llama-2 Models on HotpotQA. Here *w/ Action KB* indicates the naive KNOWAGENT and *w/o Action KB* symbolizes removing the action knowledge of the specific task.

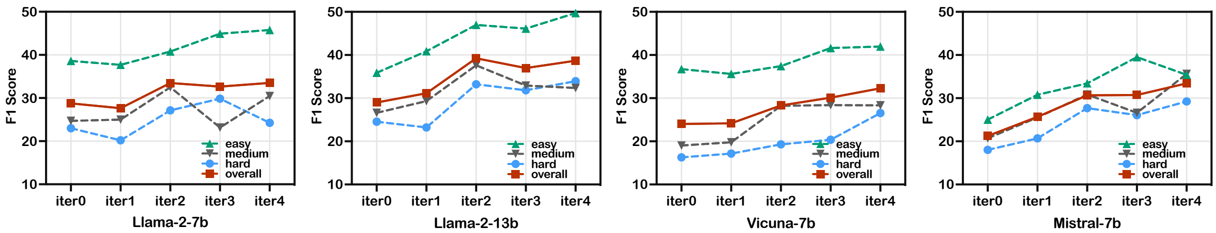

Figure 5: Ablation study on Knowledgeable Self-Learning iteration. We examine the influence of self-learning iterations on a selection of models, including Llama-2-7b, Llama-2-13b, Vicuna-7b, and Mistral-7b. Here *Iter0* represents baseline performance prior to any training.

Model	Invalid Action	Misordered Action
ReAct	2.08%	3.54%
Reflexion	6.87%	3.87%
KNOWAGENT	**0.35%**	**1.23%**

Table 2: Unreasonable action rates on HotpotQA with Llama-2-13b. Here *invalid* refers to actions that do not meet the action rule, while *misordered* means discrepancies in the logical sequence of actions.

optimization of performance. Further increasing iterations to 4 leads to an even more pronounced improvement. Consistent with previous work (Li et al., 2023b; Wu et al., 2023), these findings corroborate the efficacy of iterative self-learning in bolstering the model's comprehension of the training data, paralleling the human learning principle of "Reviewing the old as a means of realizing the new". **(2) Different base models.** We also explore other backbone models except for Llama with a 7B parameter scale, such as Vicuna-7B and Mistral-7B. The result suggests that our method is effective and generalizable across different pre-trained and fine-tuned models. However, the performance discrepancies among them also indicate a variance in the ability of different models to absorb and utilize such structured external knowledge.

Action knowledge effectively mitigates planning hallucinations. We show the statistical rates of invalid and misordered actions generated by different methods in Table 2. Here *invalid* refers to actions that do not meet the action rule, while *misordered* means discrepancies in the logical sequence of actions. Given that only the `Search` and `Finish` actions are involved in FiReAct, it is omitted from our analysis here. The results in Table 2 demonstrate that incorporating Action Knowledge significantly reduces the frequency of erroneous actions and the likelihood of invalid action paths, thereby increasing the precision of the models on the specific task. To further substantiate this claim, we refer to the experimental outcomes from KNOWAGENT and ReAct within HotpotQA, as demonstrated in Appendix 6. For a given question, ReAct's action sequence follows a `Lookup->Search->Search` pattern, which is problematic due to the dependency of the Lookup action on the subsequent Search step. However, with constraints, KNOWAGENT avoids such faulty sequences, enhancing task accuracy.

Distilled Knowledge vs. Manually Designed Knowledge. To investigate whether utilizing advanced LLMs can supplant manual efforts in constructing task-specific action knowledge, we compare distilled outcomes from `gpt-4-0613` with

KnowAgent	Strategy	HotpotQA	ALFWorld
7B	Manual	33.47	20.15
	Distilled	25.22	18.66
13B	Manual	39.26	52.24
	Distilled	25.72	51.49

Table 3: Comparative Experiment on Manual vs. Distilled Action Knowledge. *Manual* stands for human-crafted knowledge and *Distilled* represents the distilled knowledge from GPT-4.

manual-designed ones. For HotpotQA, we observe that the action knowledge distilled by GPT-4 is more concise, with fewer cyclical actions than those set by humans. This efficiency holds for simpler tasks where performance parallels human-defined tasks, while underperformance is found on more complex tasks where longer action sequences are required. For ALFWorld, the GPT-distilled action knowledge closely mirrors that crafted by humans, underscoring the model's capacity to comprehend real-world constraints. Aligning with prior research (Ding et al., 2023; Zhou et al., 2024), this distilled knowledge aids the model in understanding real-world limitations, showing little difference in effectiveness compared to human-created one.

Error Analysis. Upon analyzing the capabilities of KnowAgent, we identify its limitations, particularly in processing complex queries and summarizing extensive textual data. KnowAgent struggle to distill key information effectively, often failing to deliver accurate responses. The core issue lies in their insufficient reasoning and memory capacities for handling long contexts. Consequently, the generated responses may be incorrect or even misaligned with the posed questions, such as providing a simple `yes/no` when a specific entity is required. Future enhancements should focus on strengthening the long-text processing, information retention, and reasoning abilities of our work.

5 Related Work

LLM-Based Agents. LLM-based agents (Wang et al., 2023b; Xi et al., 2023; Durante et al., 2024) have emerged as one of the most prevalent AI systems after the rise of LLMs (Zhao et al., 2023; Qiao et al., 2023b; Zhu et al., 2023; Li et al., 2024a; Jiang et al., 2024). They learn to interact with the external world through action-observation pairs expressed by natural language. Previous works primarily focus on unlocking the potential of LLMs as the core of language agents by leveraging human-crafted (Yao et al., 2023b; Li et al., 2023a; Talebirad and Nadiri, 2023; Qian et al., 2023) or machine-generated (Zhou et al., 2023b; Chen et al., 2023c,b) prompts. Recently, there has been a growing emphasis on endowing open-source LLMs with agent capabilities through fine-tuning (Yin et al., 2023; Qiao et al., 2024; Shen et al., 2024). However, the training trajectory data for existing language agent fine-tuning methods largely rely on annotations from LLMs. This can result in the inclusion of trajectories that violate some action knowledge rules and are difficult to identify, leading to an unstable action performance of the trained language agents. Recently, Guan et al. (2024) introduce AMOR, an agent framework that constructs its reasoning capabilities on a Finite State Machine (FSM). Li et al. (2024b) introduce a "Formal-LLM" framework for agents, combining the expressiveness of natural language with the precision of formal language to enhance agent capabilities. Different from those approaches, we propose knowledge-augmented language agents that incorporate action-related knowledge rules to constrain the trajectory generation, reducing the occurrence of unreasonable action logic in the generated trajectories.

Knowledge Augmented LLMs. Previous works (Guu et al., 2020; Lewis et al., 2020; Izacard et al., 2023) concentrate on knowledge augmentation in LLMs through retrieval. Due to the rich parameterized knowledge within LLMs (Chen, 2023; Feng et al., 2023), some other works (Liu et al., 2022; Yu et al., 2023; Sun et al., 2023b) advocate for knowledge generation rather than retrieval. With the emergence of Augmented Language Models (ALMs), many studies (Trivedi et al., 2023; Li et al., 2023c; Vu et al., 2023; Qiao et al., 2023a) have enhanced the reasoning capabilities of LLMs by incorporating knowledge from external tools such as search engines, knowledge bases, and Wikipedia documents. Recent research (Zhou et al., 2023b; Ye et al., 2023) has introduced structured knowledge to regulate the workflow of LLM-based multi-agents, but none of these works has focused on restricting the action space of individual agents. To the best of our knowledge, we are the first to introduce structured action knowledge to enhance the planning capability of LLM-based agents, specifically aiming at reducing instances of planning errors caused by invalid actions during the planning process.

6 Conclusion

In this study, we introduce KNOWAGENT, a framework designed to mitigate *planning hallucinations* by incorporating external action knowledge into synthetic trajectories. Our method involves utilizing action knowledge to guide the model's action generation, translating this knowledge into text for deeper model comprehension, and employing a *knowledgeable self-learning* phase for continuous improvement. This multifaceted approach not only enhances the planning capabilities of agents but also proves effective in complex scenarios. Our experiments across various models demonstrate that KNOWAGENT effectively competes with or surpasses other baselines, showcasing the benefits of integrating external action knowledge to streamline planning processes and improve performance.

Limitations

Our limitations are listed as follows:

Task Expandability. The current experiments are conducted exclusively on the commonsense QA and household datasets. However, our approach is also applicable to a broader range of fields including medical (Tang et al., 2023), arithmetic (Cobbe et al., 2021), web browsing (Xie et al., 2023), and embodied agents (Yang et al., 2023). This suggests a potential for wider applicability that has yet to be explored.

Multi-Agent Systems. Presently, our research focuses on the application of single agents. Future studies should explore multi-agent systems, such as Chen et al. (2023c) and Qiao et al. (2024), which complete planning tasks through division of labor and collaboration. This enhancement could help agents better handle complex tasks and adapt to changing environments.

Automated Design of Action Knowledge Bases. The creation of action knowledge bases is still manual, time-consuming, and labor-intensive. Even though we use GPT-4 for distilling Action Knowledge, manual adjustments are needed. Future work should aim at automating this process to reduce manual effort and improve the model's autonomous learning and versatility.

References

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

Guangyao Chen, Siwei Dong, Yu Shu, Ge Zhang, Jaward Sesay, Börje F. Karlsson, Jie Fu, and Yemin Shi. 2023b. Autoagents: A framework for automatic agent generation. *CoRR*, abs/2309.17288.

Huajun Chen. 2023. Large knowledge model: Perspectives and challenges. *CoRR*, abs/2312.02706.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chen Qian, Chi-Min Chan, Yujia Qin, Yaxi Lu, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2023c. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *CoRR*, abs/2308.10848.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

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. *CoRR*, abs/2306.06070.

Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. 2023. Integrating action knowledge and llms for task planning and situation handling in open worlds. *Auton. Robots*, 47(8):981–997.

Zane Durante, Qiuyuan Huang, Naoki Wake, Ran Gong, Jae Sung Park, Bidipta Sarkar, Rohan Taori, Yusuke Noda, Demetri Terzopoulos, Yejin Choi, Katsushi Ikeuchi, Hoi Vo, Li Fei-Fei, and Jianfeng Gao. 2024. Agent AI: surveying the horizons of multimodal interaction. *CoRR*, abs/2401.03568.

Zhangyin Feng, Weitao Ma, Weijiang Yu, Lei Huang, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. Trends in integration of knowledge and large language models: A survey and taxonomy of methods, benchmarks, and applications. *CoRR*, abs/2311.05876.

Jian Guan, Wei Wu, Zujie Wen, Peng Xu, Hongning Wang, and Minlie Huang. 2024. AMOR: A recipe for building adaptable modular knowledge agents through process feedback. *CoRR*, abs/2402.01469.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *CoRR*, abs/2305.14909.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume

119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, and Chenglin Wu. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *CoRR*, abs/2308.00352.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. 2022. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR.

Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. 2024. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. *CoRR*, abs/2402.11163.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023a. CAMEL: communicative agents for "mind" exploration of large scale language model society. *CoRR*, abs/2303.17760.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023b. Self-alignment with instruction backtranslation. *CoRR*, abs/2308.06259.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Lidong Bing, Shafiq R. Joty, and Soujanya Poria. 2023c. Chain of knowledge: A framework for grounding large language models with structured knowledge bases. *CoRR*, abs/2305.13269.

Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. 2024a. Personal LLM agents: Insights and survey about the capability, efficiency and security. *CoRR*, abs/2401.05459.

Zelong Li, Wenyue Hua, Hao Wang, He Zhu, and Yongfeng Zhang. 2024b. Formal-llm: Integrating formal language and natural language for controllable llm-based agents. *CoRR*, abs/2402.00798.

Jiacheng Liu, Skyler Hallinan, Ximing Lu, Pengfei He, Sean Welleck, Hannaneh Hajishirzi, and Yejin Choi. 2022. Rainier: Reinforced knowledge introspector for commonsense question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 8938–8958. Association for Computational Linguistics.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2023a. Agentbench: Evaluating llms as agents. *CoRR*, abs/2308.03688.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Le Xue, Shelby Heinecke, Rithesh Murthy, Yihao Feng, Zeyuan Chen, Juan Carlos Niebles, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2023b. BOLAA: benchmarking and orchestrating llm-augmented autonomous agents. *CoRR*, abs/2308.05960.

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

Siqi Ouyang and Lei Li. 2023. Autoplan: Automatic planning of interactive decision-making tasks with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 3114–3128. Association for Computational Linguistics.

Haojie Pan, Zepeng Zhai, Hao Yuan, Yaojia Lv, Ruiji Fu, Ming Liu, Zhongyuan Wang, and Bing Qin. 2023. Kwaiagents: Generalized information-seeking agent system with large language models. *CoRR*, abs/2312.04889.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *CoRR*, abs/2307.07924.

Shuofei Qiao, Honghao Gui, Huajun Chen, and Ningyu Zhang. 2023a. Making language models better tool learners with execution feedback. *CoRR*, abs/2305.13068.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. 2023b. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5368–5393. Association for Computational Linguistics.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. AUTOACT: automatic agent learning from scratch via self-planning. *CoRR*, abs/2401.05268.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *CoRR*, abs/2302.04761.

Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. 2024. Small llms are weak tool learners: A multi-llm agent. *CoRR*, abs/2401.07324.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: language agents with verbal reinforcement learning. *CoRR*, abs/2303.11366.

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, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas Griffiths. 2024. Cognitive architectures for language agents. *Transactions on Machine Learning Research*. Survey Certification.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2023a. Adaplanner: Adaptive planning from feedback with language models. *CoRR*, abs/2305.16653.

Zhiqing Sun, Xuezhi Wang, Yi Tay, Yiming Yang, and Denny Zhou. 2023b. Recitation-augmented language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Yashar Talebirad and Amirhossein Nadiri. 2023. Multi-agent collaboration: Harnessing the power of intelligent LLM agents. *CoRR*, abs/2306.03314.

Xiangru Tang, Anni Zou, Zhuosheng Zhang, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. 2023. Medagents: Large language models as collaborators for zero-shot medical reasoning. *CoRR*, abs/2311.10537.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

XAgent Team. 2023. Xagent: An autonomous agent for complex task solving.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10014–10037. Association for Computational Linguistics.

Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. 2023. Can large language models really improve by self-critiquing their own plans? *CoRR*, abs/2310.08118.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry W. Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc V. Le, and Thang Luong. 2023. Freshllms: Refreshing large language models with search engine augmentation. *CoRR*, abs/2310.03214.

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

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2023b. A survey on large language model based autonomous agents. *CoRR*, abs/2308.11432.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. 2023. Self-evolved diverse data sampling for efficient instruction tuning. *CoRR*, abs/2311.08182.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huan, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *CoRR*, abs/2309.07864.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, Leo Z. Liu, Yiheng Xu, Hongjin Su, Dongchan Shin, Caiming Xiong, and Tao Yu. 2023. Openagents: An open platform for language agents in the wild. *CoRR*, abs/2310.10634.

Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F. Fouhey, and Joyce Chai. 2023. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent. *CoRR*, abs/2309.12311.

Ke Yang, Jiateng Liu, John Wu, Chaoqi Yang, Yi R. Fung, Sha Li, Zixuan Huang, Xu Cao, Xingyao Wang, Yiquan Wang, Heng Ji, and Chengxiang Zhai. 2024. If LLM is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents. *CoRR*, abs/2401.00812.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, Devansh Arpit, Ran Xu, Phil Mui, Huan Wang, Caiming Xiong, and Silvio Savarese. 2023c. Retroformer: Retrospective large language agents with policy gradient optimization. *CoRR*, abs/2308.02151.

Yining Ye, Xin Cong, Shizuo Tian, Jiannan Cao, Hao Wang, Yujia Qin, Yaxi Lu, Heyang Yu, Huadong Wang, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2023. Proagent: From robotic process automation to agentic process automation. *CoRR*, abs/2311.10751.

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. *CoRR*, abs/2311.05657.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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

Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, and Hai Zhao. 2023. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents. *CoRR*, abs/2311.11797.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures. *CoRR*, abs/2402.03620.

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

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, Shiding Zhu, Jiyu Chen, Wentao Zhang, Ningyu Zhang, Huajun Chen, Peng Cui, and Mrinmaya Sachan. 2023b. Agents: An open-source framework for autonomous language agents. *CoRR*, abs/2309.07870.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *CoRR*, abs/2305.13168.

A Experimental Settings

Datasets and Metrics. HotpotQA (Yang et al., 2018) is specifically designed for multi-hop reasoning tasks and comprises approximately 113,000 question-answering pairs derived from Wikipedia articles. In our experiments, a subset of 500 data instances is randomly sampled for training. For testing, we utilize the same test set as BOLAA (Liu et al., 2023b), which included 100 samples at each of three difficulty levels: easy, medium and hard. To measure performance, we adopt the F1 score as a benchmark, which also serves as the basis for the reward metric utilized in BOLAA. More details of HotpotQA are listed in Appendix B.

ALFWorld (Shridhar et al., 2021) is an interactive, text-based household environment where agent challenge to complete six different types of multi-step tasks. To train our model, we randomly select 85 instances of each task category from the training set. Furthermore, we refer to previous research (Shridhar et al., 2021; Yao et al., 2023b) and use 134 unseen validation tasks to evaluate our method. In ALFWorld, we utilize goal-conditioned success rates as our evaluation metric. We show prompts used for both datasets in Appendix D.

Baselines Our research concentrates solely on the performance of a single agent, intentionally excluding multi-agent studies from our baseline comparison. We choose the following baselines here: (1) Chain-of-Thought (CoT) (Wei et al., 2022) catalyzes in-depth reasoning in large language models (LLMs) by incorporating intermediate reasoning steps within examples. (2) ReAct (Yao et al., 2023b) enables LLMs to intertwine the generation of inferential trajectories and actions, allowing the model to better generalize and adjust its action plans. (3) The Reflexion (Shinn et al., 2023) method utilizes self-reflective feedback to promote continuous agent development through the assimilation of lessons from past errors, thus refining task execution. (4) Furthermore, FiReAct (Chen et al., 2023a) introduces a novel strategy to fine-tune LLM using diverse prompts and trajectories across tasks, demonstrating that richer fine-tuning data can further elevate the performance of agents. It's important to note that our experiments concentrate on exploring the impact of external action knowledge on LLMs, and as such, we have chosen a single-agent framework as the baseline for comparison.

Implementation. We evaluate KNOWAGENT on HotpotQA (Yang et al., 2018) and ALFWorld (Shridhar et al., 2021). We employ `Llama-2-{7,13,70}b-chat` (Touvron et al., 2023) as the backbone models, and also apply KNOWAGENT to vicuna-7b-v1.5-16k (Zheng et al., 2023) and Mistral-7B-Instruct-v0.1 (Jiang et al., 2023). During the *self-learning* phase, we set the number of training iterations to two. We fine-tune these models over 5 epochs using LoRA (Hu et al., 2022), setting a learning rate of 1×10^{-4} and the sequence length of 4096 across all scales. Additionally, we employ an Alpaca-style (Taori et al., 2023) templating approach for generating input from instructional data.

B Datasets

In HotpotQA, we have enriched the action set in previous works (Yao et al., 2023b; Liu et al., 2023b) by incorporating Bing Search as an external knowledge source. As a result, we now feature four distinct actions within HotpotQA to enhance its functionality and performance: (1) Retrieve[entity]: Retrieve the exact entity on Wikipedia and return the first paragraph if it exists. If not, return some similar entities for searching. (2) Search[topic]: Use Bing Search to find relevant information on a specified topic, question, or term. (3) Lookup[keyword]: Return the next sentence that contains the keyword in the last passage successfully found by Search or Retrieve. (4) Finish[answer]: Return the answer and conclude the task.

C Impact of Action Knowledge

See Figure 6

D Prompt

D.1 Prompt for HotpotQA

See Table 4.

D.2 Prompt for ALFWorld

See Table 5 to Table 10.

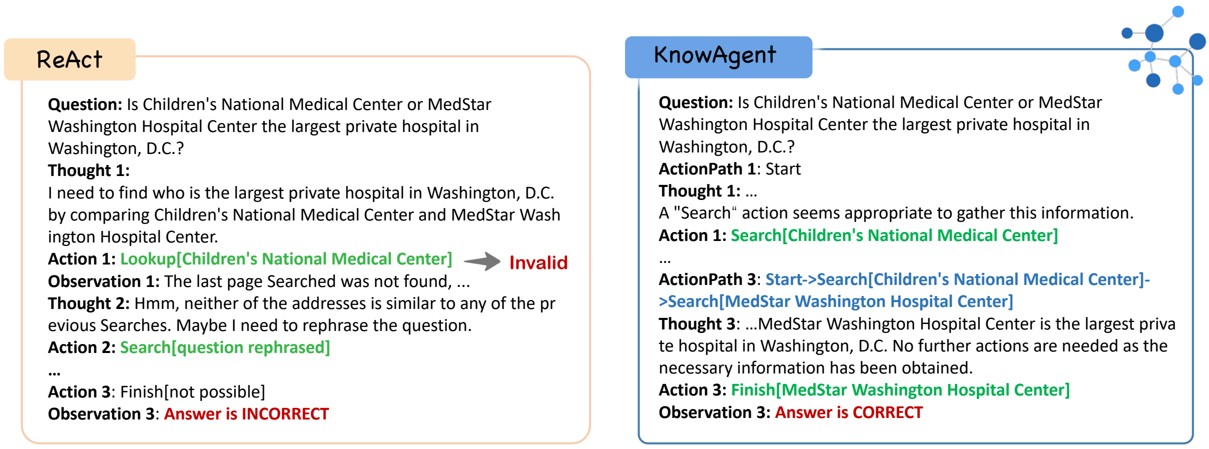

Figure 6: KNOWAGENT (the blue one) successfully completes planning, while ReAct (the orange one) exhibits *planning hallucinations* that contradict action knowledge.

Prompt for HotpotQA
Your task is to answer a question using a specific graph-based method. You must navigate from the "Start" node to the "Finish" node by following the paths outlined in the graph. The correct path is a series of actions that will lead you to the answer. The decision graph is constructed upon a set of principles known as "Action Knowledge", outlined as follows: Start:(Search, Retrieve) Retrieve:(Retrieve, Search, Lookup, Finish) Search:(Search, Retrieve, Lookup, Finish) Lookup:(Lookup, Search, Retrieve, Finish) Finish:() Here's how to interpret the graph's Action Knowledge: From "Start", you can initiate with either a "Search" or a "Retrieve" action. At the "Retrieve" node, you have the options to persist with "Retrieve", shift to "Search", experiment with "Lookup", or advance to "Finish". At the "Search" node, you can repeat "Search", switch to "Retrieve" or "Lookup", or proceed to "Finish". At the "Lookup" node, you have the choice to keep using "Lookup", switch to "Search" or "Retrieve", or complete the task by going to "Finish". The "Finish" node is the final action where you provide the answer and the task is completed. Each node action is defined as follows: (1) Retrieve[entity]: Retrieve the exact entity on Wikipedia and return the first paragraph if it exists. If not, return some similar entities for searching. (2) Search[topic]: Use Bing Search to find relevant information on a specified topic, question, or term. (3) Lookup[keyword]: Return the next sentence that contains the keyword in the last passage successfully found by Search or Retrieve. (4) Finish[answer]: Return the answer and conclude the task. As you solve the question using the above graph structure, interleave ActionPath, Thought, Action, and Observation steps. ActionPath documents the sequence of nodes you have traversed within the graph. Thought analyzes the current node to reveal potential next steps and reasons for the current situation. You may take as many steps as necessary. Here are some examples: {examples} (END OF EXAMPLES) Question: {question}{scratchpad}

Table 4: Prompt for HotpotQA.

ALFWorld - Pick

Interact with a household to solve a task by following the structured "Action Knowledge". The guidelines are:
Goto(receptacle) -> Open(receptacle)
[Goto(receptacle), Open(receptacle)] -> Take(object, from: receptacle)
Take(object, from: receptacle) -> Goto(receptacle)
[Goto(receptacle), Take(object, from: receptacle)] -> Put(object, in/on: receptacle)

Here's how to interpret the Action Knowledge:
Before you open a receptacle, you must first go to it. This rule applies when the receptacle is closed.
To take an object from a receptacle, you either need to be at the receptacle's location, or if it's closed, you need to open it first.
Before you go to the new receptacle where the object is to be placed, you should take it.
Putting an object in or on a receptacle can follow either going to the location of the receptacle or after taking an object with you.

The actions are as follows:
1) go to receptacle
2) take object from receptacle
3) put object in/on receptacle
4) open receptacle

As you tackle the question with Action Knowledge, utilize both the ActionPath and Think steps. ActionPath records the series of actions you've taken, and the Think step understands the current situation and guides your next moves.

Here are two examples.
{examples}
Here is the task.

Table 5: Prompt for the `Pick` Task.

ALFWorld - Light
Interact with a household to solve a task by following the structured "Action Knowledge". The guidelines are: [Goto(receptacle)] -> Open(receptacle) [Goto(receptacle), Open(receptacle)] -> Take(object, from: receptacle) [Goto(receptacle)] -> Use(receptacle) Here's how to interpret the Action Knowledge: Before you open a receptacle, you must first go to it. This rule applies when the receptacle is closed. To take an object from a receptacle, you either need to be at the receptacle's location, or if it's closed, you need to open it first. To use an receptacle, you must go to the place where it is located. The actions are as follows: 1) go to receptacle 2) take object from receptacle 3) use receptacle 4) open receptacle As you tackle the question with Action Knowledge, utilize both the ActionPath and Think steps. ActionPath records the series of actions you've taken, and the Think step understands the current situation and guides your next moves. Here are two examples. {examples} Here is the task.

Table 6: Prompt for the `Light` Task.

ALFWorld - Clean
Interact with a household to solve a task by following the structured "Action Knowledge". The guidelines are: [Goto(receptacle)] -> Open(receptacle) [Goto(receptacle), Open(receptacle)] -> Take(object, from: receptacle) [Goto(receptacle), Take(object, from: receptacle)] -> Put(object, in/on: receptacle) [Put(object, from: receptacle)] -> Clean(object, with: receptacle) Here's how to interpret the Action Knowledge: Before you open a receptacle, you must first go to it. This rule applies when the receptacle is closed. To take an object from a receptacle, you either need to be at the receptacle's location, or if it's closed, you need to open it first. Putting an object in or on a receptacle can follow either going to the location of the receptacle or after taking an object with you. To clean an object using a receptacle, the object must first be placed in or on that receptacle. The actions are as follows: 1) go to receptacle 2) take object from receptacle 3) open receptacle 4) put object in/on receptacle 5) clean object with receptacle As you tackle the question with Action Knowledge, utilize both the ActionPath and Think steps. ActionPath records the series of actions you've taken, and the Think step understands the current situation and guides your next moves. Here are two examples. {examples} Here is the task.

Table 7: Prompt for the `Clean` Task.

ALFWorld - Heat
Interact with a household to solve a task by following the structured "Action Knowledge". The guidelines are: [Goto(receptacle)] -> Open(receptacle) [Goto(receptacle), Open(receptacle)] -> Take(object, from: receptacle) [Goto(receptacle), Take(object, from: receptacle)] -> Put(object, in/on: receptacle) [Put(object, in/on: receptacle)] -> Heat(object, with: receptacle) Here's how to interpret the Action Knowledge: Before you open a receptacle, you must first go to it. This rule applies when the receptacle is closed. To take an object from a receptacle, you either need to be at the receptacle's location, or if it's closed, you need to open it first. Putting an object in or on a receptacle can follow either going to the location of the receptacle or after taking an object with you. To heat an object using a receptacle, the object must first be placed in or on that receptacle. The actions are as follows: 1) go to receptacle 2) take object from receptacle 3) open receptacle 4) put object in/on receptacle 5) heat object with receptacle As you tackle the question with Action Knowledge, utilize both the ActionPath and Think steps. ActionPath records the series of actions you've taken, and the Think step understands the current situation and guides your next moves. Here are two examples. {examples} Here is the task.

Table 8: Prompt for the `Heat` Task.

ALFWorld - Cool
Interact with a household to solve a task by following the structured "Action Knowledge". The guidelines are: [Goto(receptacle)] -> Open(receptacle) [Goto(receptacle), Open(receptacle)] -> Take(object, from: receptacle) [Goto(receptacle), Take(object, from: receptacle)] -> Put(object, in/on: receptacle) [Put(object, in/on: receptacle)] -> Cool(object, with: receptacle) Here's how to interpret the Action Knowledge: Before you open a receptacle, you must first go to it. This rule applies when the receptacle is closed. To take an object from a receptacle, you either need to be at the receptacle's location, or if it's closed, you need to open it first. Putting an object in or on a receptacle can follow either going to the location of the receptacle or after taking an object with you. To cool an object using a receptacle, the object must first be placed in or on that receptacle. The actions are as follows: 1) go to receptacle 2) take object from receptacle 3) open receptacle 4) put object in/on receptacle 5) cool object with receptacle As you tackle the question with Action Knowledge, utilize both the ActionPath and Think steps. ActionPath records the series of actions you've taken, and the Think step understands the current situation and guides your next moves. Here are two examples. {examples} Here is the task.

Table 9: Prompt for the `Cool` Task.

ALFWorld - Pick Two
Interact with a household to solve a task by following the structured "Action Knowledge". The guidelines are: Goto(receptacle) -> Open(receptacle) [Goto(receptacle), Open(receptacle)] -> Take(object, from: receptacle) Take(object, from: receptacle) -> Goto(receptacle) [Goto(receptacle), Take(object, from: receptacle)] -> Put(object, in/on: receptacle) Here's how to interpret the Action Knowledge: Before you open a receptacle, you must first go to it. This rule applies when the receptacle is closed. To take an object from a receptacle, you either need to be at the receptacle's location, or if it's closed, you need to open it first. Before you go to the new receptacle where the object is to be placed, you should take it. Putting an object in or on a receptacle can follow either going to the location of the receptacle or after taking an object with you. Ensure the first object is placed before proceeding to deposit the second object. The actions are as follows: 1) go to receptacle 2) take object from receptacle 3) put object in/on receptacle 4) open receptacle As you tackle the question with Action Knowledge, utilize both the ActionPath and Think steps. ActionPath records the series of actions you've taken, and the Think step understands the current situation and guides your next moves. Here are two examples. {examples} Here is the task.

Table 10: Prompt for the `Pick Two` Task.