Agent Planning with World Knowledge Model

Shuofei Qiao[♠*], **Runnan Fang**[♠*], **Ningyu Zhang**[♠†], **Yuqi Zhu**[♠], **Xiang Chen**[♠], **Shumin Deng**[♣], **Yong Jiang**[♢], **Pengjun Xie**[♢], **Fei Huang**[♢], **Huajun Chen**[♠†]

[♠]Zhejiang University

[♣]National University of Singapore [♢]Alibaba Group

{shuofei,zhangningyu}@zju.edu.cn

Abstract

Recent endeavors towards directly using large language models (LLMs) as agent models to execute interactive planning tasks have shown commendable results. Despite their achievements, however, they still struggle with brainless trial-and-error in global planning and generating hallucinatory actions in local planning due to their poor understanding of the real world. Imitating humans' *mental* world knowledge model which provides global prior knowledge before the task and maintains local dynamic knowledge during the task, in this paper, we introduce *parametric* **W**orld **K**nowledge **M**odel (**WKM**) to facilitate agent planning. Concretely, we steer the agent model to self-synthesize knowledge from both expert and sampled trajectories. Then we develop a WKM, providing prior *task knowledge* to guide the global planning and dynamic *state knowledge* to assist the local planning. Experimental results on three real-world simulated datasets with Mistral-7B, Gemma-7B, and Llama-3-8B demonstrate that our method can achieve superior performance compared to various strong baselines. Besides, we analyze to illustrate that our WKM can effectively alleviate the blind trial-and-error and hallucinatory action issues, providing strong support for the agent's understanding of the world[1].

1 Introduction

The remarkable advances in Large Language Models (LLMs) have witnessed a rapid development of various natural language processing tasks (Meta, 2024; Jiang et al., 2023; OpenAI, 2023; Touvron et al., 2023; Zhao et al., 2023a; Qiao et al., 2023b). Recently, multiple attempts that directly exploit LLMs as agent models to carry out physical world planning tasks have demonstrated promising achievements (Yao et al., 2023; Zeng et al., 2023; Yin et al., 2023; Qiao et al., 2024; Shen et al., 2024; Zhu et al., 2024; Song et al., 2024). However, as most state-of-the-art LLMs are autoregressive models trained with next-token prediction, they lack the ability to essentially understand the real world, leading to generating hallucinatory actions and performing brainless trial-and-error in the environment as shown in Figure 1(a).

In contrast to LLMs, humans possess a mental knowledge model about the physical world (Briscoe, 2011; Johnson-Laird, 1983, 2010; Pramod et al., 2020). When facing a specific task, they will first briefly rehearse the entire process in mind using their rich prior knowledge before performing mindless actions. We call this kind of knowledge global *task knowledge*. In addition, during the task procedure, the mental world knowledge model will constantly maintain a kind of local *state knowledge*, representing humans' cognition of the current world state. For example, imagine you are in a room and your task is to `put a clean egg in microwave`. The *task knowledge* may refer to `The egg is most likely in the fridge` ... `The workflows are: 1) locate and take the egg; 2) clean the egg using sinkbasin` ... The *state knowledge* possibly refers to `My task is to` ... `I have found and taked the egg` ... `Next I should` ... The absence of world knowledge can lead to blind trial-and-error in the early planning stages when environmental information is limited. Conversely, in later stages when information is redundant, it can easily result in a confused cognition of the current world state and generate hallucinatory actions.

The process by which humans handle planning tasks reminds us to develop a parametric **W**orld **K**nowledge **M**odel (**WKM**) to facilitate agent model planning. As humans typically acquire knowledge from expertise and practical experience,

[*] Equal contribution.

[†] Corresponding Author.

[1]Code will be available at https://github.com/zjunlp/WKM.

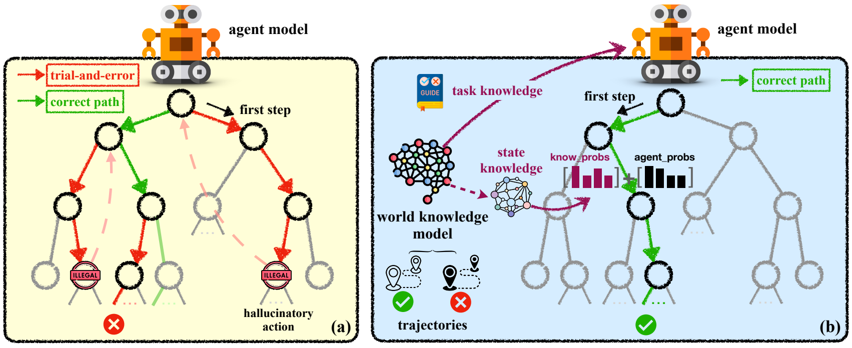

Figure 1: Traditional agent planning vs. Agent planning with world knowledge model.

we first enable the agent model to self-synthesize task knowledge from the comparison between expert trajectories and self-exploration trajectories. Then we prompt the agent model to summarize state knowledge for each planning step from expert trajectories and combine the previous and next actions to build a state knowledge base. Lastly, we integrate the generated knowledge into expert trajectories and train a world knowledge model. The agent model needs to be re-trained to adapt to the incorporation of task knowledge. Note our agent model and knowledge model are both trained with LoRA (Hu et al., 2022) sharing the same backbone.

During the planning phase, we use the WKM to provide global prior task knowledge and maintain local dynamic state knowledge for the agent model as shown in Figure 1(b). The task knowledge will be concatenated in natural language form following the specific task to guide the agent model's trial-and-error. At each planning step, to prevent the occurrence of hallucinatory actions, we utilize the generated state knowledge as the query to conduct kNN retrieval from the pre-built state knowledge base. We then use the constraints from the previous action, the probabilities of the retrieved next actions, and the probabilities from the agent model to make a weighted prediction for the next action.

We evaluate our method on three real-world simulated planning tasks with three state-of-the-art open-source LLMs: Mistral-7B (Jiang et al., 2023), Gemma-7B (Mesnard et al., 2024), and Llama-3-8B (Meta, 2024). Empirical results demonstrate that our method achieves superior performance compared to various strong baselines on both seen and unseen tasks. Moreover, further analytical results show that 1) our WKM can effectively reduce blind trial-and-error and hallucinatory actions, 2) our model-generated instance-level knowledge can generalize better to unseen tasks, 3) weak-guide-strong is feasible, 4) multi-task unified WKM possesses strong potential, and 5) explicit state knowledge will hurt the performance of agent planning.

2 Preliminaries

We mainly focus on interactive tasks with partial observations from environments. Following the task formulation in (Song et al., 2024), the problem can be viewed as a Partially Observable Markov Decision Process (POMDP): $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T})$. The instruction space $\mathcal{U}$ defines the task and its corresponding regulations. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $\mathcal{O}$ is the observation space. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ defines the transition function, which we assume to be given. It is noticed that $\mathcal{U}$, $\mathcal{A}$, and $\mathcal{O}$ are subspaces of the natural language space in the language agent scenarios.

Based on the above, the historical trajectory h_t that consists of a list of actions and observations at time t can be represented as:

$$h_t = (u, a_0, o_0, a_1, o_1, \dots, a_t, o_t), \tag{1}$$

where $u \in \mathcal{U}$ is the task instruction and $a \in \mathcal{A}$, $o \in \mathcal{O}$ are the action and the observation. Given a task, the language agent with parameter θ serves as the policy model π_θ responsible for generating the action a_{t+1} based on h_t at each time step $t+1$:

$$a_{t+1} \sim \pi_\theta(\cdot|h_t). \tag{2}$$

Specifically, $a_0 \sim \pi_\theta(\cdot|u)$ is generated according to the task instruction u. The whole trajectory τ concludes when the task is completed or exceeds the maximum time steps. Then the entire trajectory with time length n can be modeled as:

$$\pi_\theta(\tau|u) = \prod_{t=0}^{n} \pi_\theta(a_{t+1}|h_t)\pi_\theta(a_0|u). \tag{3}$$

Ultimately, the final reward $r(u, \tau) \in [0, 1]$ is calculated. Note that we follow a ReAct-style (Yao et al., 2023) trajectory that includes rationales before each action. We use a to represent the action with rationales for convenience.

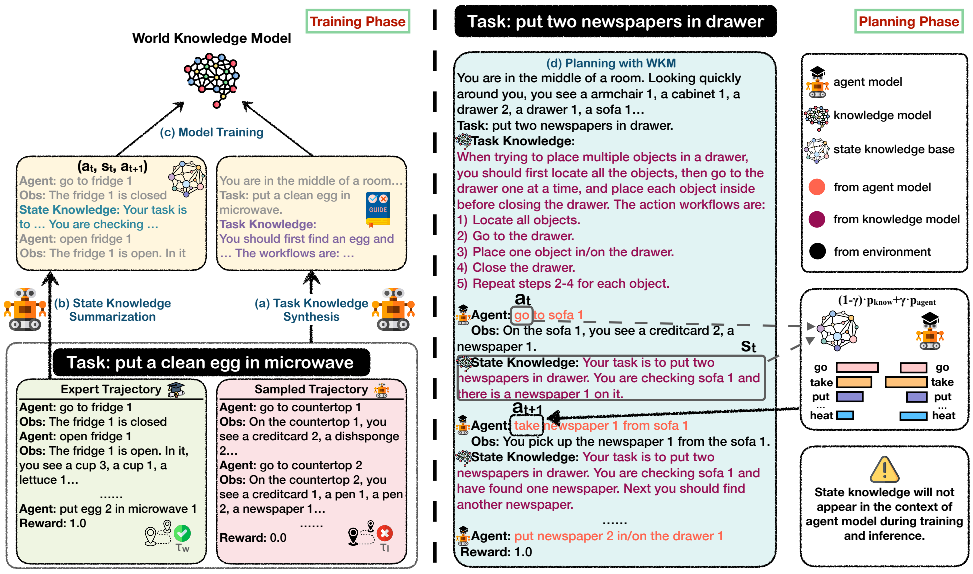

Figure 2: **Overview of our WKM**. We train a world knowledge model on the knowledge synthesized by the agent model itself from both expert and explored trajectories, providing prior task knowledge to guide global planning and dynamic state knowledge to assist local planning.

3 Method

We introduce agent planning with world knowledge model, of which the world knowledge consists of two components: *task knowledge* and *state knowledge*, as shown in Figure 2

3.1 Task Knowledge Synthesis

The *task knowledge* serves as the prior knowledge to guide the agent model's global planning and prevent it from dropping into blind trial-and-error.

Experienced Agent Exploration. We primarily acquire task knowledge through the comparison of preference trajectories (*chosen* vs. *rejected*). In order to improve the quality of rejected trajectories and obtain more targeted task knowledge, we employ an experienced agent for exploration. Firstly, we train a vanilla language model with expert trajectories[2] from the training set to obtain an experienced agent. Subsequently, the experienced agent explores the training set tasks again to generate rejected trajectories. Our purpose is to extract superior task knowledge that cannot be acquired solely through supervised fine-tuning on chosen trajectories, thus further effectively boosting the agent's capabilities.

Self Knowledge Synthesis. With the expert trajectories as the chosen ones and the trajectories sampled from the experienced agent as the rejected ones, we prompt the agent model itself to synthesize the task knowledge. Supposing $\mathcal{K}$ is the task knowledge space:

$$\kappa \sim \pi_\theta(\cdot|\rho_{\text{TaskKnow}}, u, \tau_w, \tau_l), \tag{4}$$

where $\kappa \in \mathcal{K}$ is the task knowledge, ρ_{TaskKnow} stands for the prompt to instruct the task knowledge extraction, and τ_w, τ_l are the chosen and rejected trajectories respectively. Note that given the same task u, τ_w and τ_l always satisfy $r(u, \tau_w) \geq r(u, \tau_l)$. Even when $r(u, \tau_w) = r(u, \tau_l)$, we still consider trajectories sampled from the experienced agent as rejected ones. This is because expert trajectories often have shorter step lengths, enabling the agent to learn more knowledge of efficient planning. For detailed prompts of task knowledge synthesis, please refer to Appendix H.1.

3.2 State Knowledge Summarization

The *state knowledge* serves as the dynamic knowledge to constrain the agent model's local planning and prevent it from generating hallucinatory actions. We prompt the agent model to self-summarize state knowledge at each planning step based on the expert trajectories to guarantee quality. For detailed prompts of state knowledge summarization, please refer to Appendix H.2. Supposing the prompt used to summarize state knowledge is $\rho_{\text{StateKnow}}$ and the state knowledge $s \in \mathcal{S}$ is a part of the state space $\mathcal{S}$, the generation of state knowledge at time t can be represented as:

$$s_t \sim \pi_\theta(\cdot|\rho_{\text{StateKnow}}, h_t). \tag{5}$$

[2]For details on collecting expert trajectories, please refer to Appendix A

State Knowledge Base Construction. To avoid confusion caused by excessive additional information, instead of explicitly concatenating the state knowledge to the context, we construct a state knowledge base for retrieval (we analyze in §4.3 how explicit state knowledge may affect the performance of agent model). We combine the state knowledge s_t with the previous action a_t and next action a_{t+1} from the expert trajectory to form a action-state-action triplet (a_t, s_t, a_{t+1}). After iterating through all expert trajectories, we obtain a State Knowledge Base $\mathcal{B} = \{(s, a_{\text{pre}}, a_{\text{next}})^{(i)}\}_{i=1}^{|\mathcal{B}|}$, where $|\mathcal{B}|$ is the size of the state knowledge base.

3.3 Model Training

We integrate the generated world knowledge into expert trajectories and train a world knowledge model. The agent model needs to be re-trained to adapt to the incorporation of task knowledge. Note that our agent model and knowledge model are both trained with LoRA sharing the same backbone. We list the examples of training data for both the agent model and WKM in Appendix E.

Agent Model Training. Given the expert trajectories dataset $\mathcal{D} = \{(u, \kappa, \tau_w)^{(i)}\}_{i=1}^{|\mathcal{D}|}$ with task knowledge κ generated in §3.1, we train the agent model to follow the task knowledge to generate actions. Under an auto-regressive manner, the loss of the agent model can be formulated as:

$$\mathcal{L}_{\text{agent}}(\pi_\theta) = -\mathbb{E}_{\tau_w \sim \mathcal{D}}[\pi_\theta(\tau_w|u, \kappa)] \tag{6}$$

Suppose $\mathcal{X} = (x_1, x_2, \ldots, x_{|\mathcal{X}|})$ is the token sequence of the trajectory τ_w, we have:

$$\pi_\theta(\tau_w|u, \kappa) = -\sum_{j=1}^{|\mathcal{X}|} \left(\mathbb{1}(x_j \in \mathcal{A}) \times \log \pi_\theta(x_j|u, \kappa, x_{<j})\right). \tag{7}$$

Here $\mathbb{1}(x_j \in \mathcal{A})$ is the indicator function to mask tokens related to observations.

World Knowledge Model Training. The main difference in the training data between the agent and knowledge model is the added state knowledge. Given the expert trajectories dataset with both task and state knowledge $\mathcal{D}' = \{(u, \kappa, \tau'_w)^{(i)}\}_{i=1}^{|\mathcal{D}'|}$ where $\tau'_w = (a_0, o_0, s_0, \ldots, a_n, o_n, s_n)$, the loss of the knowledge model π_ϕ can be formulated as:

$$\mathcal{L}_{\text{know}}(\pi_\phi) = -\mathbb{E}_{\kappa, \tau'_w \sim \mathcal{D}'}[\pi_\phi(\kappa|u)\pi_\phi(\tau'_w|u, \kappa)] \tag{8}$$

Suppose $\mathcal{X}' = (x'_1, x'_2, \ldots, x'_{|\mathcal{X}'|})$ is the token sequence of the expert trajectory with state knowledge τ'_w and $\mathcal{Y} = (y_1, y_2, \ldots, y_{|\mathcal{Y}|})$ represents the token sequence of the task knowledge κ, we have:

$$\pi_\phi(\kappa|u) = -\sum_{i=1}^{|\mathcal{Y}|} \log \pi_\phi(y_i|u, y_{<i}) \tag{9}$$

$$\pi_\phi(\tau'_w|u, \kappa) = -\sum_{j=1}^{|\mathcal{X}'|} \left(\mathbb{1}(x'_j \in \mathcal{S}) \times \log \pi_\phi(x'_j|u, \kappa, x'_{<j})\right), \tag{10}$$

where $\mathbb{1}(x_j \in \mathcal{S})$ is the indicator function to mask tokens unrelated to state knowledge.

3.4 Agent Planning with World Knowledge Model

At inference time, the agent model plans on the evaluation tasks with the aid of the world knowledge model. We redefine the historical trajectory $h_t = (u, \kappa, a_0, o_0, a_1, o_1, \ldots, a_t, o_t)$. Given a specific task instruction u, the knowledge model first generates the task knowledge $\kappa \sim \pi_\phi(\cdot|u)$, then the agent model starts planning. Assuming the available action set $\mathcal{A}_u \subseteq \mathcal{A}$ for the task u is $(\alpha_u^{(1)}, \alpha_u^{(2)}, \ldots, \alpha_u^{(|\mathcal{A}_u|)})$, at any time $t \geq 0$, instead of directly generating a next action $a_{t+1} \in \mathcal{A}_u$ based on h_t, we first employ the world knowledge model to generate the current state knowledge $s_t \sim \pi_\phi(\cdot|h_t)$ and leverage s_t to query the state knowledge base $\mathcal{B} = \{(s, a_{\text{pre}}, a_{\text{next}})^{(i)}\}_{i=1}^{|\mathcal{B}|}$. With the state knowledge as the `key`, we retrieve $\mathcal{N}$ nearest triplets **from where** $a_{\text{pre}} = a_t$ based on semantic similarity and collect the corresponding next actions a_{next}. We count the probability of each action $p_{\text{know}}(\alpha_u^{(i)}) = \frac{\mathcal{N}_i}{\mathcal{N}}$, where $\mathcal{N}_i$ is the occurrence number of action $\alpha_u^{(i)}$ in all the collected a_{next}. Therefore, we get the probability acquired from the state knowledge base:

$$P_{\text{know}}(\mathcal{A}_u) = (p_{\text{know}}(\alpha_u^{(1)}), p_{\text{know}}(\alpha_u^{(2)}), \cdots, p_{\text{know}}(\alpha_u^{(|\mathcal{A}_u|)})), \quad \sum_{i=1}^{|\mathcal{A}_u|} p_{\text{know}}(\alpha_u^{(i)}) = 1. \tag{11}$$

Backbone	Method	ALFWorld		WebShop	ScienceWorld	
		Seen	Unseen		Seen	Unseen
GPT-3.5-Turbo	REACT	8.57	5.97	44.37	15.41	13.99
GPT-4		44.29	38.05	62.76	67.32	65.09
Mistral-7B	REACT	7.86	5.22	14.63	20.72	17.65
	Reflexion	11.56	6.00	16.64	21.07	18.11
	NAT	64.43	68.96	61.01	57.12	50.79
	ETO	66.84	<u>71.43</u>	<u>64.09</u>	58.17	<u>51.85</u>
	KNOWAGENT	<u>70.44</u>	70.72	61.28	<u>59.32</u>	47.24
	WKM	**73.57** +3.13	**76.87** +5.44	**65.48** +1.39	**62.12** +2.80	**53.62** +1.77
Gemma-7B	REACT	6.43	2.24	5.93	3.58	3.51
	Reflexion	7.14	2.99	7.71	4.94	3.93
	NAT	67.86	65.88	55.82	47.63	44.98
	ETO	66.43	<u>68.66</u>	<u>62.67</u>	<u>50.44</u>	<u>47.84</u>
	KNOWAGENT	<u>69.29</u>	67.60	58.80	48.55	45.28
	WKM	**70.71** +1.42	**70.40** +1.74	**63.75** +1.08	**53.68** +3.24	**49.24** +1.40
Llama-3-8B	REACT	2.86	3.73	19.32	24.76	22.66
	Reflexion	4.29	4.48	22.73	27.23	25.41
	NAT	60.71	59.70	61.60	55.24	48.76
	ETO	64.29	<u>64.18</u>	<u>64.57</u>	57.90	<u>52.33</u>
	KNOWAGENT	<u>66.71</u>	62.69	64.40	<u>58.67</u>	49.18
	WKM	**68.57** +1.86	**65.93** +1.75	**66.64** +2.07	**60.12** +1.55	**54.75** +2.42

Table 1: **Main Results.** The best results of each model are marked in **bold** and the second-best results are marked with <u>underline</u>. All the prompt-based baselines ( ) are evaluated under one-shot prompting and all the fine-tuning-based baselines ( ) are trained through LoRA. Red represents the changes of WKM relative to the optimal results in the baselines.

Afterward, we sample all the logits of $\alpha_u^{(i)}, 1 \leq i \leq |\mathcal{A}_u|$ from the agent model and apply a `softmax` function to normalize the probability. We define the probability acquired from the agent model as:

$$P_{\text{agent}}(\mathcal{A}_u) = (p_{\text{agent}}(\alpha_u^{(1)}), p_{\text{agent}}(\alpha_u^{(2)}), \cdots, p_{\text{agent}}(\alpha_u^{(|\mathcal{A}_u|)})), \quad \sum_{i=1}^{|\mathcal{A}_u|} p_{\text{agent}}(\alpha_u^{(i)}) = 1. \tag{12}$$

Finally, we determine the next action by combining the above two probabilities:

$$a_{t+1} = \underset{\alpha_u^{(i)} \in \mathcal{A}_u, 1 \leq i \leq |\mathcal{A}_u|}{\arg\max} (\gamma \cdot p_{\text{agent}}(\alpha_u^{(i)}) + (1-\gamma) \cdot p_{\text{know}}(\alpha_u^{(i)})), \tag{13}$$

where γ is the hyperparameter that controls the proportion of $P_{\text{agent}}(\mathcal{A}_u)$.

4 Experiments

4.1 Experimental Settings

Datasets and Metrics. We evaluate our method on three real-world simulated planning datasets: **ALFWorld** (Shridhar et al., 2021), **WebShop** (Yao et al., 2022), and **ScienceWorld** (Wang et al., 2022). AlFWorld and ScienceWorld include unseen tasks to evaluate the agent's generalization ability. The reward of ALFWorld is binary 0 or 1, indicating whether the agent has completed the task or not. WebShop and ScienceWorld provide dense rewards from 0 to 1 to measure the completion level of the task. For all the datasets, we apply **average reward** as the final metrics. Please refer to Appendix B for detailed dataset information.

Models and Baselines. We evaluate on three state-of-the-art open-source models: 1) **Mistral-7B** (Jiang et al., 2023), the Mistral-7B-Instruct-v0.2 version. 2) **Gemma-7B** (Mesnard et al., 2024), the Gemma-1.1-7B-it version. 3) **Llama-3-8B** (Meta, 2024), the Meta-Llama-3-8B-Instruct version. We compare our method with two prompt-based baselines: **REACT** (Yao et al., 2023) and **Reflexion** (Shinn et al., 2023). Besides, we adopt two strong baselines that introduce rejected trajectories into the training process to learn from experience: **NAT** (Wang et al., 2024b), learn from rejected trajectories through SFT, and **ETO** (Song et al., 2024), learn from rejected trajectories through DPO (Rafailov et al., 2023). Moreover, we compare with a knowledge-augmented planning method **KNOWAGENT**. We also include **ChatGPT** (gpt-3.5-turbo-0125) (OpenAI, 2022) and **GPT-4** (gpt-4-32K) for comparison. All the prompt-based baselines are tested under one-shot and all the fine-tuning-based baselines are trained with LoRA (Hu et al., 2022). Please refer to Appendix C for base-

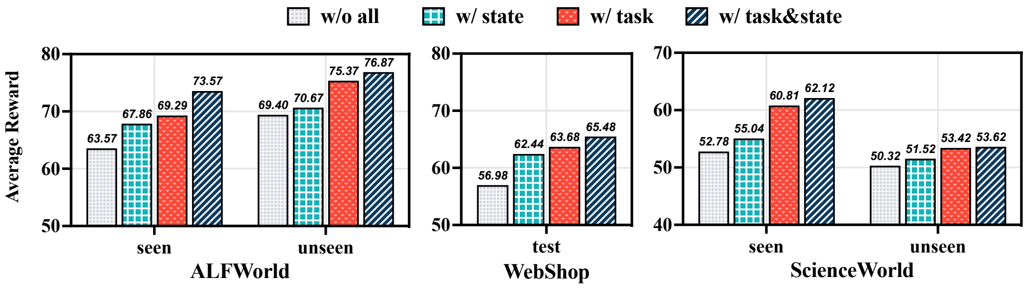

Figure 3: **Ablation Study** on Mistral-7B. **w/o all** means the vanilla experienced agent model training with pure expert trajectories. **w/ state** is testing agent model with only state knowledge base constraints. **w/ task** stands for guiding agent model with only task knowledge. **w/ task&state** is our WKM with both task knowledge guidance and state knowledge constraints.

lines and re-producing details.

Training and Inference Setups. We fine-tune all our models with LoRA (Hu et al., 2022) using the LlamaFactory (Zheng et al., 2024) framework. We set the learning rate of 1e-4 and the sequence length of 2048 for all the models. The training epoch is 3 and the batch size is 32. We use the AdamW optimizer (Loshchilov and Hutter, 2019) with a cosine learning scheduler. During inference, the number of retrieved action-state-action triplets $\mathcal{N}$ is set to 3000 and the $P_{\text{agent}}(\mathcal{A}_u)$ weight γ is set to $\{0.4, 0.5, 0.7\}$. All the training and inference experiments are conducted on 8 NVIDIA V100 32G GPUs within 12 hours. Please refer to Appendix D for detailed hyperparameters used in our paper.

4.2 Results

Main Results. As shown in Table 1, **for prompt-based baselines** on open-source models, both ReAct and Reflexion exhibit poor performance, far behind our method and fine-tuning-based baselines on various datasets. GPT-3.5-Turbo performs ordinarily on two datasets other than WebShop, and it even falls behind Mistral-7B and Llama-3-8B's ReAct performance on ScienceWorld. However, GPT-4 exhibits strong performance across various datasets. Nevertheless, our approach, through LoRA training alone, surpasses GPT-4 on ALFWorld and WebShop. **For fine-tuning-based baselines**, both NAT and ETO fall behind our method, implying that just integrating world knowledge for agent models is worth more than further fussy SFT or DPO on negative examples. Our method also performs better than KnowAgent which brings human-designed fixed action knowledge and long action paths into trajectories. This suggests the effectiveness of our WKM which is responsible for generating instance-level task knowledge and maintaining implicit action constraints.

Approach Ablations. As shown in Figure 3, taking Mistral-7B as an example, we decompose the key components of WKM to examine the roles of the task and state knowledge separately. In a macro view, removing each module results in a clear drop in the agent's performance, which validates the power of our world knowledge. Furthermore, the improvement through task knowledge (*w/ task*) is more pronounced than that through state knowledge (*w/ state*), suggesting the necessity of global prior knowledge for agent planning. A more micro observation reveals that the impact of state knowledge is more significant on seen tasks compared to unseen tasks, while the influence of task knowledge is sustainable across seen and unseen tasks. This may be attributed that although our real-time state knowledge is generated by WKM, the state knowledge base is built on the training set, which may weaken generalization to some extent.

4.3 Analysis

World knowledge can mitigate blind trial-and-error and reduce hallucinatory actions. We compare the number of planning steps for each dataset and calculate the average steps of each method. As depicted in Figure 10 (in Appendix), WKM demonstrates the ability to complete a significant proportion of tasks using the shortest trajectory, indicating that guidance from world knowledge can effectively reduce the agent's blind trial-and-error. Taking a further perspective from an average standpoint in Table 2, it can be observed that WKM exhibits lower average planning steps compared to other baselines. As ALFWorld can respond to invalid actions, in Table 3, we count the percentage of hallucinatory actions that occurred in trajectories from ALFWorld for each method. The results confirm the effectiveness of our world knowledge model to decrease hallucinatory actions. Furthermore, it is worth noting that most baselines

Method	ALFWorld		WebShop	ScienceWorld	
	Seen	Unseen		Seen	Unseen
NAT	23.27	23.42	4.08	20.18	21.21
ETO	19.82	22.29	3.99	24.13	26.35
KNOWAGENT	18.51	24.56	4.01	21.06	24.74
WKM	**17.66**	**17.92**	**3.97**	**18.74**	**19.59**

Table 2: **Average Steps.** The maximum number of steps in ALFWorld and WebShop is 40 and 10. In ScienceWorld, the number of steps ranges from 10 to 120 depending on the task type, with an average of around 40.

Method	ALFWorld	
	Seen	Unseen
NAT	45.71%	50.00%
ETO	34.29%	36.57%
KNOWAGENT	33.57%	44.78%
WKM	**32.86%**	**29.85%**

Table 3: **Hallucinatory Action Rates** on ALFWorld. We count the rates of trajectories containing invalid actions regardless of their correctness.

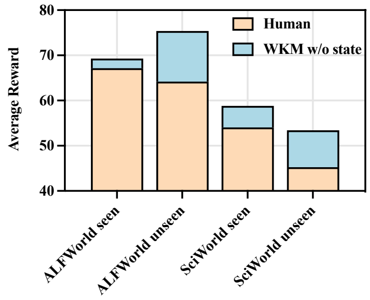

Figure 4: Performance of human-designed dataset-level knowledge vs. WKM generated instance-level knowledge.

Backbone	Method	ALFWorld	
		Seen	Unseen
GPT-3.5-Turbo	REACT	8.57	5.97
	WKM w/o state	**12.86**	**8.96**
GPT-4	REACT	44.29	38.05
	WKM w/o state	**50.71**	**47.01**

Table 4: **Weak-guide-strong**. The knowledge model here is based on Mistral-7B.

show a prominent increase in the average number of steps and percentage of invalid actions when transitioning from seen tasks to unseen tasks, but WKM can still maintain a relatively low level. This reflects laterally that our world knowledge can still effectively guide the agent model on unseen tasks, highlighting the knowledge generalization brought by the world knowledge model. To see how our world knowledge works, please refer to our case study in Appendix F.

Our instance-level knowledge can generalize better to unseen tasks. To further explore the benefit of using a knowledge model to generate instance-level task knowledge, we carefully survey the task knowledge generated by our WKM and abstract it into dataset-level knowledge for each dataset. Then we retrain the agent model to adapt to new dataset-level knowledge[3]. As illustrated in Figure 4, we compare the performance of dataset-level knowledge with our instance-level task knowledge (WKM *w/o state*) on ALFWorld and ScienceWorld. It can be observed that our model-generated instance-level knowledge not only surpasses human-designed knowledge on seen tasks but also exhibits even more remarkable performance on unseen tasks, with the improvement in performance on unseen tasks significantly greater than that on seen tasks. This phenomenon straightly reflects the strong generalization ability of our knowledge model compared to rigidly designed knowledge by humans.

Weak knowledge model guides strong agent model planning. In our main experiments, the knowledge model and agent model are based on the same backbone. Here, we explore on ALFWorld what will happen if we use a weak knowledge model to guide a strong agent model. We choose Mistral-7B as the backbone of the knowledge model and ChatGPT and GPT-4 as the agent model. Since we cannot get the token distribution from OpenAI API, we only apply task knowledge to the agent model. As exhibited in Table 4, the results of both ChatGPT and GPT-4 show distinct advances after being guided by the Mistral-7B world knowledge model, indicating the weak world knowledge model also contains knowledge that the strong model may lack. This inspires us with a new agent learning paradigm: **weak-guide-strong**. Due to its lightweight nature, the weak knowledge model can flexibly adjust its parameters based on the needs of the agent model, which can address the difficulty of large agent models in adapting to new environments through fine-tuning.

Unified World Knowledge Model Training. We mix the world knowledge collected from all three datasets and jointly train one single world knowledge model to investigate the effect of multi-task world knowledge learning. Figure 5 illustrates the relative performance comparison between multi-task WKM and various baselines, from which we can observe that multi-task WKM not only does not lead to performance degradation but also ex-

[3]Detailed manually designed dataset-level knowledge prompt can be found in Appendix H.3

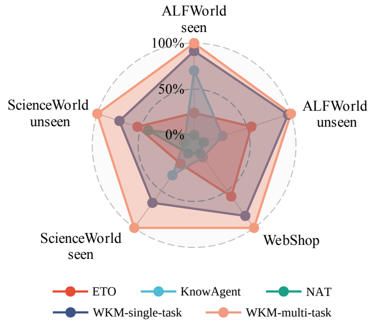

Figure 5: Relative performance of multi-task WKM compared to various baselines.

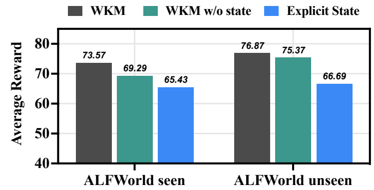

Figure 6: Performance of explicit state knowledge.

hibits visible improvements compared to single-task WKM, especially on WebShop and ScienceWorld. Similar to (Zeng et al., 2023; Zhang et al., 2024; Chen et al., 2024) which endeavor to train a unified agent model and achieve strong generalization ability to held-out tasks, this observation inspires us with the potential of training a unified world knowledge model that can be applied to help various held-in agent models and also generalize to guide held-out agent models. A more daring idea is whether a unified agent model combined with a unified world knowledge model is the key to Artificial General Intelligence (AGI).

Explicit state knowledge will hurt the planning performance. To demonstrate the rationality of constructing a state knowledge base, we explore the effect of incorporating state knowledge into the context of the agent model (we retrain the agent model to follow both the task and state knowledge), as shown in Figure 6. The performance of explicit state knowledge is far inferior to our approach of retrieving from a state knowledge base and utilizing probabilistic constraints. It even performs worse than when we remove state knowledge and only include task knowledge. This clearly indicates that blindly extending prompts with a large amount of explicit natural language feedback is lose-more-than-gain for agent planning, and implicit knowledge constraints may be sometimes more prudent.

5 Related Work

LLM Agents. LLMs have emerged as a promising avenue towards unlocking the potential of Artificial General Intelligence, offering robust support for the development of agent systems (Wang et al., 2024a; Xi et al., 2023; Guo et al., 2024; Zhou et al., 2023). Existing research in this domain primarily focuses on agent planning (Huang et al., 2022; Logeswaran et al., 2022; Yao et al., 2023; Song et al., 2023a), external tools harnessing (Shen et al., 2023; Lu et al., 2023; Song et al., 2023b; Patil et al., 2023; Qiao et al., 2023a; Qin et al., 2023; Tang et al., 2023), code generation (Sun et al., 2023; Logeswaran et al., 2022; Qian et al., 2023; Hong et al., 2023), etc. Recently, there has been a growing focus on endowing open-source LLMs with agent functionalities through fine-tuning (Chen et al., 2023; Zeng et al., 2023; Yin et al., 2023; Shen et al., 2024; Song et al., 2024; Wang et al., 2024b). However, these approaches rely on blindly fitting the probabilities of tokens to learn planning, without having an intimate cognition of the environment. The lack of knowledge can lead to the agent blindly attempting trial-and-error and generating hallucinatory actions.

Knowledge Augmented Agent Planning. Planning (Huang et al., 2024) is a crucial capability for intelligent agents to accomplish real-world tasks, often requiring agents to possess rich knowledge and environmental commonsense. Few works have explored the field of knowledge-augmented agent planning. (Huang et al., 2022; Zhao et al., 2023b; Ding et al., 2023) utilize the rich parametric knowledge stored in pre-trained language models to assist agent planners. (Guan et al., 2024; Li et al., 2024; Zhao et al., 2024; Zhu et al., 2024) design structured or natural language knowledge to regulate the actions. However, the above studies require the manual design of fixed prompt templates or task procedures, making it challenging to transfer across different task environments. (Zhou et al., 2023; Ye et al., 2023; Fu et al., 2024) propose the automation of knowledge generation using language models. However, their knowledge either consists of only global workflow or only local action principles. In contrast, we train our world knowledge model both on global task knowledge and local state knowledge to assist agent planning, and these knowledge sources are derived from the model's self-summary rather than hand-curated.

6 Conclusion

In this paper, we strive to develop a parametric world knowledge model (WKM) to augment language agent model planning. Our WKM can generate prior task knowledge to guide global planning as well as dynamic state knowledge to regulate local planning. Our extensive results show that our world knowledge can work on both GPT-4 and state-of-the-art open-source models and achieve superior performance compared to various strong baselines. Analytical experiments validate that our WKM can 1) reduce brainless trial-and-error and invalid actions, 2) generalize better to unseen tasks, 3) achieve weak-guide-strong, and 4) be effectively extended to unified world knowledge training.

Limitations

Despite our best efforts, this paper may still have some limitations: 1) Our primary intention behind designing the WKM is to compensate for the lack of world knowledge in the agent model. However, determining what a language model knows and doesn't know has been an ongoing challenge that remains unresolved. 2) It is widely acknowledged that world knowledge extends beyond textual representations. While our world knowledge is currently limited to textual information, exploring multi-modal world knowledge models is indeed one of our important future tasks. 3) Our world knowledge model cannot dynamically update with the changes of the world and feedback from the agent. 4) Generating world knowledge can introduce additional inference overhead.

Ethics Statement

This research was conducted with the highest ethical standards and best practices in research. All our experiments use publicly available datasets (as detailed in Appendix B), avoiding ethical concerns related to privacy, confidentiality, or misuse of personal biological information. However, despite our best efforts, it is not avoidable if someone maliciously modifies the world knowledge model to contradict the world's knowledge and leads the agent to engage in unethical behavior.

References

Robert Eamon Briscoe. 2011. Mental imagery and the varieties of amodal perception. *Pacific Philosophical Quarterly*, 92(2):153–173.

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

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024. Agent-flan: Designing data and methods of effective agent tuning for large language models. *CoRR*, abs/2403.12881.

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.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024. Autoguide: Automated generation and selection of state-aware guidelines for large language model agents. *CoRR*, abs/2403.08978.

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.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *CoRR*, abs/2402.01680.

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.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9118–9147. PMLR.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of LLM agents: A survey. *CoRR*, abs/2402.02716.

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.

Philip N Johnson-Laird. 2010. Mental models and human reasoning. *Proceedings of the National Academy of Sciences*, 107(43):18243–18250.

Philip Nicholas Johnson-Laird. 1983. *Mental models: Towards a cognitive science of language, inference, and consciousness*. Harvard University Press.

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

Lajanugen Logeswaran, Yao Fu, Moontae Lee, and Honglak Lee. 2022. Few-shot subgoal planning with language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5493–5506. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, and et al. 2024. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. `https://ai.meta.com/blog/meta-llama-3/`.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. `https://openai.com/blog/chatgpt/`.

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

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *CoRR*, abs/2305.15334.

RT Pramod, Michael Cohen, Kirsten Lydic, Josh Tenenbaum, and Nancy Kanwisher. 2020. Evidence that the brain's physics engine runs forward simulations of what will happen next. *Journal of Vision*, 20(11):1521–1521.

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.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *CoRR*, abs/2307.16789.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving AI tasks with chatgpt and its friends in hugging face. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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.

Chan Hee Song, Brian M. Sadler, Jiaman Wu, Wei-Lun Chao, Clayton Washington, and Yu Su. 2023a. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 2986–2997. IEEE.

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

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

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. 2023. Adaplanner: Adaptive planning from feedback with language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, and Le Sun. 2023. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *CoRR*, abs/2306.05301.

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, and et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

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 Jirong Wen. 2024a. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345.

Renxi Wang, Haonan Li, Xudong Han, Yixuan Zhang, and Timothy Baldwin. 2024b. Learning from failure: Integrating negative examples when fine-tuning large language models as agents. *CoRR*, abs/2402.11651.

Ruoyao Wang, Peter A. Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11279–11298. Association for Computational Linguistics.

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.

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, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. 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.

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.

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.

Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Juntao Tan, Thai Hoang, Liangwei Yang, Yihao Feng, Zuxin Liu, Tulika Awalgaonkar, Juan Carlos Niebles, Silvio Savarese, Shelby Heinecke, Huan Wang, and Caiming Xiong. 2024. Agentohana: Design unified data and training pipeline for effective agent learning. *CoRR*, abs/2402.15506.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: LLM agents are experiential learners. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial*

Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, pages 19632–19642. AAAI Press.

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. 2023a. A survey of large language models. *CoRR*, abs/2303.18223.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2023b. Large language models as commonsense knowledge for large-scale task planning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *CoRR*, abs/2403.13372.

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. 2023. Agents: An open-source framework for autonomous language agents. *CoRR*, abs/2309.07870.

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

A Expert Trajectories Collection

We mainly use the expert trajectories with a ReAct-style (Yao et al., 2023) collected from (Song et al., 2024):

1. **ALFWorld** (Shridhar et al., 2021). The dataset provides human-annotated trajectories.
2. **WebSahop** (Yao et al., 2022). Except for human-annotated trajectories, GPT-4 is also applied to explore in the environment and trajectories with a reward greater than 0.7 are reserved.
3. **ScienceWorld** (Wang et al., 2022). The dataset provides heuristic searching algorithms to generate golden trajectories for each sub-task.

Since the original golden trajectories do not contain rationales, GPT-4 is further leveraged to generate the corresponding information.

B Dataset Information

We evaluate our method on three real-world simulated agent planning datasets: ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), and ScienceWorld (Wang et al., 2022).

1. **ALFWorld** is a household dataset requiring the agent to navigate through the room and manipulate objects. Except for seen tasks, AlFWorld also includes unseen tasks to evaluate the agent's generalization ability. The reward of ALFWorld is binary 0 or 1, indicating whether the agent has completed the task or not.
2. **WebShop** is an online shopping dataset in a website environment. It provides dense final rewards from 0 to 1 to measure the completion level of the task.
3. **ScienceWorld** is a scientific reasoning dataset at the level of a standard elementary school science curriculum. It also possesses both seen and unseen parts and a dense reward function from 0 to 1.

For all the datasets, we apply **average reward** as the final metrics. Table 5 illustrates the statistics of each dataset.

Table 5: Dataset statistics.

Dataset	Train	Text-Seen	Text-Unseen
ALFWorld	3,119	140	134
WebShop	1,824	200	-
ScienceWorld	1,483	194	211

C Compared Baselines

Here we detailedly introduce the baselines we compare with and our re-produce details.

1. **ReAct** (Yao et al., 2023). The first approach incorporates Chain-of-Thought (CoT) prompting in agent planning tasks with a format of Thought-Action-Observation loop. In our paper, we apply one-shot prompting for ReAct[4].
2. **Reflexion** (Shinn et al., 2023). A strong prompt-based baseline reinforces agent planning with verbal feedback. Manually designed prompts are used to enable the agent to reflect on the historical trajectory and re-plan based on the feedback. In our paper, we utilize one-shot prompting for reflection and select the first reflect iteration as our result due to limited context[5].

[4] https://github.com/ysymyth/ReAct

[5] https://github.com/noahshinn/reflexion

3. **NAT** (Wang et al., 2024b). NAT includes negative trajectories by employing different prompts during agent fine-tuning. When evaluating, only positive prompts are used to encourage the language agent to generate correct trajectories. As it also follows the REACT-style format, we directly use the default positive and negative prompts and train with LoRA in our paper[6].

4. **ETO** (Song et al., 2024). Another baseline includes negative trajectories during agent training. The method contains two training phases, of which the first phase is behavior cloning which fine-tunes the agent on expert trajectories, and the second phase is learning from failures which further fine-tunes the agent through Direct Preference Optimization (DPO) (Rafailov et al., 2023). In our paper, we remove the one-shot prompt for fairness and retain all the default hyperparameters proposed in ETO except for LoRA training[7].

5. **KNOWAGENT** (Zhu et al., 2024). KNOWAGENT is a knowledge-augmented agent planning baseline that applies action knowledge in the prompt and maintains an action path in the context during planning to constrain the agent's action. We directly use the default prompt mentioned in KNOWAGENT for ALFWorld and carefully extend it to WebShop and ScienceWorld by following a similar format[8].

All the prompt-based baselines are tested under one-shot and all the fine-tuning-based baselines are trained with LoRA (Hu et al., 2022).

D Hyperparameters

The detailed hyperparameters we use during training and inference are shown in Table 6. We employ identical hyperparameters for different models and datasets.

Table 6: Detailed hyperparameters used in our paper.

Name	**Value**
lora r	8
lora alpha	16
lora dropout	0.05
lora target modules	q_proj, v_proj
cutoff len	2048
epochs	3
batch size	32
batch size per device	4
gradient accumulation steps	2
learning rate	1e-4
warmup ratio	0.03
temperature	0.0, 0.5
retrieved state knowledge $\mathcal{N}$	3000
$P_{\text{agent}}(\mathcal{A}_u)$ weight γ	0.4, 0.5, 0.7

E Training Data Example

We show the training data example for agent model and world knowledge model in Figure 7 and Figure 8. During training, transparent tokens do not contribute to the calculation of the loss. Only the parts of the model responsible for output (agent model: purple, knowledge world model: green), participate in the loss calculation.

[6] https://github.com/Reason-Wang/NAT
[7] https://github.com/Yifan-Song793/ETO
[8] https://github.com/zjunlp/KnowAgent

Training Data Example for Agent Model

Task Instruction: You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 1, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a dresser 1, a garbagecan 1, a laundryhamper 1, and a shelf 1. Your task is to: examine the book with the desklamp.
Task Knowledge: When you cannot find the object required for the task under the desklamp, first check all the drawers and shelves in the room, then revisit the previous locations and double-check before giving up and looking in the garbage can or laundry hamper. The action workflows are as follows: 1. For each drawer and shelf, go to the location and open it to check if the object is inside. 2. If the object is not found, close the drawer or shelf and move on to the next one. 3. After checking all the drawers and shelves, revisit the previous locations and double-check if the object might have been overlooked. 4. If the object is still not found, look in the garbage can or laundry hamper as a last resort.
Thought: The task requires me to examine a book with the desklamp. First, I need to locate the book.
Action: go to bed 1
Observation: On the bed 1, you see a book 1, a laptop 1, and a pillow 1.
Thought: I see a book on the bed. I should pick it up to examine it with the desklamp.
Action: take book 1 from bed 1
Observation: You pick up the book 1 from the bed 1.
Thought: Now that I have the book, I need to find the desklamp to examine the book under its light.
Action: go to desk 1
Observation: On the desk 1, you see an alarm clock 1, a bowl 1, a CD 2, a mug 3, a mug 2, a mug 1, a pen 1, and a pencil 2.
Thought: The desklamp isn't on the desk. I need to look around the room to find where the desklamp is located.
Action: go to dresser 1
Observation: On the dresser 1, you see a cellphone 1, a desklamp 1, and a keychain 2.
Thought: I've found the desklamp on the dresser. I should use it to examine the book.
Action: use desklamp 1

Figure 7: Training Data Example for Agent Model.

Training Data Example for Agent Model

Task Instruction: You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 1, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a dresser 1, a garbagecan 1, a laundryhamper 1, and a shelf 1. Your task is to: examine the book with the desklamp.
Task Knowledge: When you cannot find the object required for the task under the desklamp, first check all the drawers and shelves in the room, then revisit the previous locations and double-check before giving up and looking in the garbage can or laundry hamper. The action workflows are as follows: 1. For each drawer and shelf, go to the location and open it to check if the object is inside. 2. If the object is not found, close the drawer or shelf and move on to the next one. 3. After checking all the drawers and shelves, revisit the previous locations and double-check if the object might have been overlooked. 4. If the object is still not found, look in the garbage can or laundry hamper as a last resort.
Thought: The task requires me to examine a book with the desklamp. First, I need to locate the book.
Action: go to bed 1

Observation: On the bed 1, you see a book 1, a laptop 1, and a pillow 1.
State Knowledge: Your task is to examine a book with the desklamp. You have not found the desklamp yet, and have only checked the bed 1.
Thought: I see a book on the bed. I should pick it up to examine it with the desklamp.
Action: take book 1 from bed 1
Observation: You pick up the book 1 from the bed 1.
State Knowledge: Your task is to examine a book using the desklamp. Your have located the book on bed 1 and picked it up.
Thought: Now that I have the book, I need to find the desklamp to examine the book under its light.
Action: go to desk 1
Observation: On the desk 1, you see an alarm clock 1, a bowl 1, a CD 2, a mug 3, a mug 2, a mug 1, a pen 1, and a pencil 2.
State Knowledge: Your task is to examine a book with the desklamp. You have located the book on bed 1 and picked it up. Next, You need to find the desklamp to examine the book under its light.
Thought: The desklamp isn't on the desk. I need to look around the room to find where the desklamp is located.
Action: go to dresser 1
Observation: On the dresser 1, you see a cellphone 1, a desklamp 1, and a keychain 2.
State Knowledge: Your task is to examine a book with the desklamp. You have located the book on the bed and picked it up, now you find a desklamp on a dresser.

Figure 8: Training Data Example for World Knowledge Model.

F Case Study

In Figure 9, we list the trajectories of ETO and our WKM within the same task in ALFWorld to illustrate how world knowledge functions. **The rationales before each action have been omitted to guarantee a clear illustration.** The task is to `clean some soapbar and put it in cabinet`. Initially, ETO blindly searches for the `soapbar` in the `countertop` and `cabinet`, introducing a lot of irrelevant information and unnecessary context. In the later stages of planning, ETO experiences the hallucination and executes the `put` action after `close` the cabinet, causing the environment to become unrecognizable and resulting in a collapse. On the contrary, guided by task knowledge, WKM directly identified the possible locations of the `soapbar` and successfully found it in the first attempt. Subsequently, WKM efficiently completed the task with precision, adhering to the constraints of state knowledge.

Case Study of WKM

ETO

Task Instruction: You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder 2, a handtowelholder 1, a sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1, and a towelholder 1. Your task is to: clean some soapbar and put it in cabinet.
 # Blind trial-and-error begins.
Agent: go to countertop 1
Observation: On the countertop 1, you see a candle 3, a candle 1, a soapbottle 1, a spraybottle 2, and a toiletpaper 2.
Agent: go to cabinet 1
Observation: The cabinet 1 is closed.
Agent: open cabinet 1
Observation: You open the cabinet 1. The cabinet 1 is open. In it, you see a soapbottle 2.
Agent: close cabinet 1
Observation: You close the cabinet 1.
 # Blind trial-and-error ends.
Agent: go to toilet 1
Observation: On the toilet 1, you see a soapbar 1, and a spraybottle 3.
Agent: take soapbar 1 from toilet 1
Observation: You pick up the soapbar 1 from the toilet 1.
Agent: go to sinkbasin 1
Observation: On the sinkbasin 1, you see nothing.
Agent: clean soapbar 1 with sinkbasin 1
Observation: You clean the soapbar 1 using the sinkbasin 1.
Agent: go to cabinet 1
Observation: The cabinet 1 is closed.
Agent: open cabinet 1
Observation: You open the cabinet 1. The cabinet 1 is open. In it, you see a soapbottle 2.
Agent: close cabinet 1
Observation: You close the cabinet 1.
 # Hallucinatory action.
Agent: put soapbar 1 in/on cabinet 1
Observation: Nothing happens.
Agent: check cabinet 1
Observation: Nothing happens.
Agent: go to exit
Observation: Nothing happens.
Agent: exit
Observation: Nothing happens.
......
Reward: 0.0

WKM

Task Instruction: ...
Task Knowledge: When the task is to clean and place an object, you should first locate the object. The possible locations of the soapbar are: toilet, sinbasin. Once the object is found, you should clean it using the sinkbasin, then put it in the cabinet. The action workflows are as follows: 1. Go to the place where the object is located. 2. Take the object from the place. 3. Clean the object using the sinkbasin. 4. Go to the cabinet. 5. Put the object in/on the cabinet.
Agent: go to toilet 1
Observation: On the toilet 1, you see a soapbar 1, and a spraybottle 3.
State Knowledge: Your task is to clean some soapbar and put it in cabinet. Now you are checking toilet 1 and there is a soapbar 1.
Agent: take soapbar 1 from toilet 1
Observation: You pick up the soapbar 1 from the toilet 1.
State Knowledge: Your task is to clean some soapbar and put it in cabinet. You have checked the toilet 1 and found the soapbar 1.
Agent: go to sinkbasin 1
Observation: On the sinkbasin 1, you see nothing.
State Knowledge: Your task is to clean some soapbar and put it in cabinet. You have checked the toilet 1 and found the soapbar 1. Now you are at the sinkbasin.
Agent: clean soapbar 1 with sinkbasin 1
Observation: You clean the soapbar 1 using the sinkbasin 1.
State Knowledge: Your task is to clean some soapbar and put it in cabinet. You have cleaned the soapbar 1 using the sinkbasin 1.
Agent: go to cabinet 1
Observation: The cabinet 1 is closed.
State Knowledge: Your task is to clean some soapbar and put it in cabinet. You have cleaned the soapbar 1 using the sinkbasin 1. Now you are checking cabinet 1.
Agent: open cabinet 1
Observation: You open the cabinet 1. In it, you see a soapbottle 2.
State Knowledge: Your task is to clean some soapbar and put it in cabinet. You have cleaned the soapbar 1 and opened the cabinet 1.
Agent: put soapbar 1 in/on cabinet 1
Reward: 1.0

Figure 9: Case study for our WKM.

G Win Rate of Planning Steps

See Figure 10.

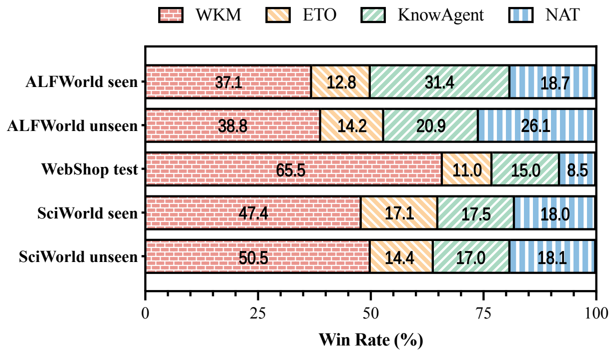

Figure 10: **Win Rate of Planning Steps.** We choose the method with the shortest steps for each task and calculate the proportion.

H Prompts

In this section, we illustrate all the prompts used in our paper.

H.1 Task Knowledge Synthesis Prompt

Prompt for Task Knowledge Synthesis

Task Knowledge

Prompt for Synthesis: I will provide you with an analysis of both a successful trajectory and an explored trajectory for the same task. By comparing the two, we can identify the key factors that contribute to success. Based on this analysis, you need to generate task-related task knowledge to help increase the success rate of future endeavors.
Success Trajectory: **Success_T**
Explored Trajectory: **Explored_T**
The task knowledge should specify what to do in what task. Here is a task knowledge example:
Task Knowledge Example
You should make your answer concise. Put your answer in this format: Task Knowledge: When ... you should (or should not) ... The action workflows are: ...

Figure 11: Prompt for Task Knowledge Synthesis.

H.2 State Knowledge Summarization Prompt

Prompt for State Knowledge Synthesis

State Knowledge

Prompt for Synthesis: You'll get a segment of a trajectory of a text-based task task. Your task is to generate a brief and general state knowledge of the now task state following "State Knowledge: ". Keep it wise and general for the same task. Here is an example:
State Knowledge Example

Now it's your turn. Here is the trajectory :
Trajectory
Make sure your output is within 128 tokens.
Put your answer in this format: State Knowledge: ...

Figure 12: Prompt for State Knowledge Summarization.

H.3 Dataset-Level Knowledge Prompt

Task Knowledge example

Alfworld Task Knowledge example

When picking an object, heat it, and place it, you should first go to the possible locations of the object, then take the object, heat it with microwave, and put it in place.
The action workflows are as follows:
1) go to receptacle
2) take object from receptacle
3) heat object with receptacle
4) go to the place to put the object
5) put object in/on receptacle

Webshop Task Knowledge example

When looking for an object you want to buy, you should first search with relevant keywords tailored to the product you are looking for, and then click the relevant tag to view the product details, if the description matches the characteristics of the target item, click[buy now].
The action workflows are as follows:
1) search with keywords or examples, if you are searching for a laptop, you might search[laptop, 14-inch, Intel Core i7]
2) click the most relevant tag to view the detailed product page.
3) check the product details one by one, like color, size, type, and price, and make sure the price is within budget.
4) if find the right items, click[buy now] to buy it.

Sciworld Task Knowledge example

When tasked with boiling apple juice, focus on locating the kitchen first. Then, locate the apple juice in the fridge. Activate the stove, pour the apple juice into a metal pot, and move the metal pot to the stove. Monitor the stove until the apple juice reaches a boiling point. Once boiled, remove the pot from the stove.
The action workflows are:
1) teleport to the kitchen.
2) look around to find the apple juice in the fridge.
3) activate the stove.
4) pour apple juice into a metal pot.
5) move the metal pot to the stove.
6) look at stove.
7) examine apple juice to confirm boiling.
8) repeat step 6,7 until apple juice is boiled.

Figure 13: Dataset-Level Task Knowledge Examples.