Agentic Knowledgeable Self-awareness

Shuofei Qiao[♠*], **Zhisong Qiu**[♠*], **Baochang Ren**[♠], **Xiaobin Wang**[♢], **Xiangyuan Ru**[♠], **Ningyu Zhang**[♠†], **Xiang Chen**[♣], **Yong Jiang**[♢†], **Pengjun Xie**[♢], **Fei Huang**[♢], **Huajun Chen**[♠♡†]

[♠]Zhejiang University [♢]Alibaba Group

[♣]Nanjing University of Aeronautics and Astronautics

[♡]Zhejiang Key Laboratory of Big Data Intelligent Computing

{shuofei,zhangningyu,huajunsir}@zju.edu.cn

Abstract

Large Language Models (LLMs) have achieved considerable performance across various agentic planning tasks. However, traditional agent planning approaches adopt a "flood irrigation" methodology that indiscriminately injects gold trajectories, external feedback, and domain knowledge into agent models. This practice overlooks the fundamental human cognitive principle of situational self-awareness during decision-making—the ability to dynamically assess situational demands and strategically employ resources during decision-making. We propose **agentic knowledgeable self-awareness** to address this gap, a novel paradigm enabling LLM-based agents to autonomously regulate knowledge utilization. Specifically, we propose **`KnowSelf`**, a data-centric approach that applies agents with **`know`**ledgeable **`self`**-awareness like humans. Concretely, we devise a heuristic situation judgement criterion to mark special tokens on the agent's self-explored trajectories for collecting training data. Through a two-stage training process, the agent model can switch between different situations by generating specific special tokens, achieving optimal planning effects with minimal costs. Our experiments demonstrate that `KnowSelf` can outperform various strong baselines on different tasks and models with minimal use of external knowledge[1].

1 Introduction

Remarkable advances in Large Language Models (LLMs) have catalyzed breakthroughs in agent-based planning systems (Xi et al., 2023; Wang et al., 2024a; Huang et al., 2024; Durante et al., 2024; Liu et al., 2025). According to how agents learn decision-making, current agent learning methods can be categorized into three types: *i)* direct trajectory imitation (Yao et al., 2023; Chen et al., 2023; Zeng et al., 2023); *ii)* trial-and-error refinement (Shinn et al., 2023; Xiang et al., 2023; Song et al., 2024b; Zhang et al., 2024a); *iii)* knowledge-augmented planning (Zhao et al., 2024a; Fu et al., 2024; Zhu et al., 2024; Chen et al., 2024).

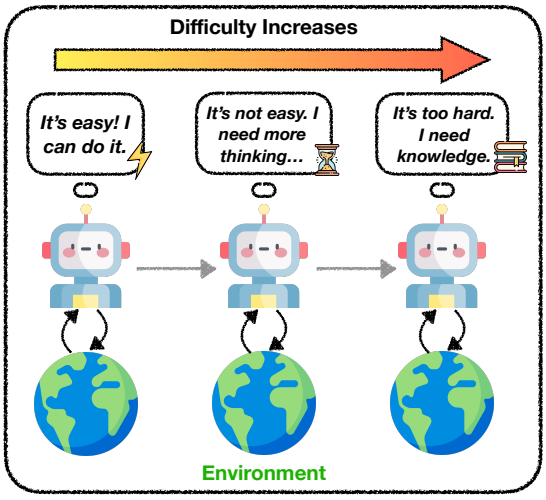


Figure 1: **Agentic Knowledgeable Self-awareness.**

However, current agent learning resembles more of an unconscious pattern-fitting process (Mirzadeh et al., 2024; Shi et al., 2023; Dziri et al., 2023). Agent models are compelled to learn implicit planning capabilities by being indiscriminately fed explicit planning trajectories, leading to a fragility towards unexpected signals during the inference process, thereby easily dropping into pattern collapse. Further enhanced approaches such as the introduction of external feedback or knowledge often tend to be a "flood irrigation" strategy, disregarding the agents' real necessity. However, excessive trial-and-error and blind incorporation of knowledge are usually unfeasible in practical settings and markedly elevate the inference cost of the model.

Conversely, self-awareness is a critical component of human decision-making (Keenan et al., 2011; Lewis et al., 2011; Lou et al., 2017). It al-

[*] Equal Contribution.

[†] Corresponding Author.

[1]Code is available at https://github.com/zjunlp/KnowSelf.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12601–12625
July 27 - August 1, 2025

lows individuals to assess their cognitive states and adapt their strategies according to dynamic external situations. This metacognitive ability enables humans to recognize when they can rely on their own abilities, when they need self-reflection, or when they need additional knowledge, thus optimizing their decision-making processes. On the contrary, current language agents lack this self-awareness capability, often leading to inefficient and brittle planning behaviors. So ***can language agents also have situational self-awareness like humans?***

In this paper, we introduce the problem of **agentic knowledgeable self-awareness** which refers to the *agent's cognition of whether itself has the ability to provide the correct next action given the current environmental situation*. To tackle this problem, we propose **`KnowSelf`**, a data-driven method that endows agent models with the ability of knowledgeable self-awareness which enables agent models to selectively introduce knowledge based on the current situation in the environment (see Figure 1). Specifically, we enable the agent to self-explore and gather different situations within the environment. Then we design a heuristic criterion to classify three kinds of situations (*fast thinking, slow thinking, knowledgeable thinking*) and mark them with special tokens to produce self-awareness training data. Subsequently, a two-stage training process is employed to train the agent model's self-awareness capability. We first conduct supervised fine-tuning to teach agent models initial self-awareness planning patterns. Then we utilize an RPO loss (Pang et al., 2024) to further boost self-awareness abilities. Finally, the agent signifies its situational awareness by generating certain special tokens, enabling selective reflection or knowledge incorporation during inference.

We evaluate `KnowSelf` on two simulated agent planning datasets with two different scales of models. Experimental results show that `KnowSelf` can achieve superior performance with minimal reflection and knowledge compared to various baselines. Moreover, we conduct further analysis to examine the scaling law, generalization, and mechanism of agentic knowledgeable self-awareness. In a nutshell, our contributions are as follows:

- **Problem and Method.** We propose agentic knowledgeable self-awareness and introduce `KnowSelf` to enable agent models to selectively query knowledge based on situations.
- **Experiments.** Experimental results show that `KnowSelf` can achieve optimal performance with minimal reflection and knowledge compared to various baselines.
- **Analysis.** Except for ablation studies, we further explore the scaling law, generalization, and mechanism of agentic self-awareness.

2 Background

A dynamic interactive environment can be regarded as a Partially Observable Markov Decision Process: $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O})$. Initially, a specific task $u \in \mathcal{U}$ is typically accompanied by an initial environmental state $s_0 \in \mathcal{S}$. Given the current state s, after performing an action $a \in \mathcal{A}$, the state transition function $T(s'|s, a) \in \mathcal{T}$ determines the next state s'. Due to partial observation, the current state is provided to the language agent in the form of an observation $o \in \mathcal{O}$. Then the historical interaction trajectory at time t can be represented as $h_t = (u, a_0, o_0, a_1, o_1, \ldots, a_t, o_t)$. In our scenario, a language agent π backed by an LLM with parameters θ is responsible for deciding the next action a_{t+1} based on the historical trajectory h_t:

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

Most current methods rely on fitting Equation 1 to make decisions, which is more akin to rote memorization. So in this paper, we propose **agentic knowledgeable self-awareness**. Please note that the self-awareness mentioned here differs from the previous concept of LLMs' knowledge boundary (Cheng et al., 2024; Yin et al., 2024; Wen et al., 2024). The focus here is on the agent's self-awareness in dynamic situations, rather than on static factual knowledge. Specifically, we define three types of situations based on agents' ability:

- **Fast thinking.** The agent is able to directly provide the correct action with little thinking.
- **Slow thinking.** The agent is able to provide the correct action but requires multiple steps of thinking and reflection.
- **Knowledgeable thinking.** The agent is unable to provide the correct action and needs to rely on external knowledge for thinking.

We go beyond the paradigm of fast or slow thinking (Yu et al., 2024; Saha et al., 2024; Chen et al., 2025; Li et al., 2025) and further introduce external

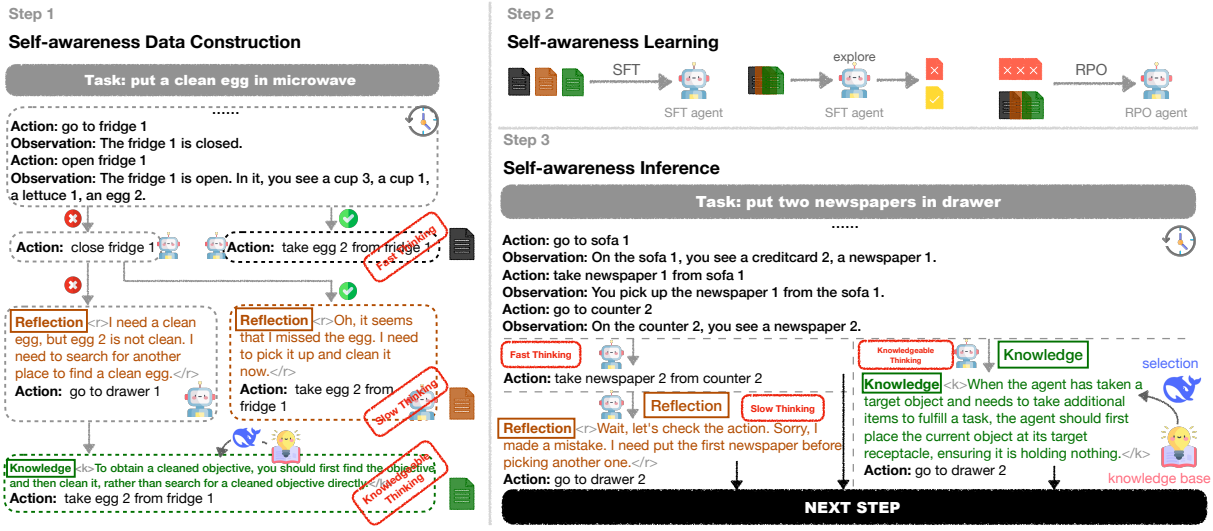


Figure 2: **The framework of our `KnowSelf`.** Firstly, we mark self-explored trajectories with special tokens according to the situation judgement criterion to form the training data. Secondly, we apply a two-stage training framework to teach the agent model knowledgeable self-awareness abilities. Finally, the agent model identifies different situations by generating specific special tokens during inference.

knowledge into the thinking system of LLMs, striving to enhance the knowledgeable self-awareness ability of language agents.

3 Method

3.1 Knowledge System Construction

Given that our emphasis is on knowledgeable self-awareness rather than the construction of a knowledge system, we draw upon and polish up a simple yet effective knowledge collection method outlined in Chen et al. (2024) to minimize costs in this process. The formation of the knowledge base is offline and lightweight, relying on an extremely minimal number of trajectories to be completed. A detailed knowledge system construction process can be found in Appendix A. We denote the final knowledge system as $\mathcal{S} : (\mathcal{K}, \mathcal{R})$, where $\mathcal{K} = \{k_1, k_2, ..., k_{\mathcal{N}_{\max}}\}$ is the knowledge base and $\mathcal{R}$ is the knowledge selection module that can select the required knowledge based on the agent's historical trajectory h_t. Please note that here our concept of "knowledge" can encompass various forms of knowledge sources, such as symbolic knowledge, parameterized knowledge, knowledge obtained through web searches, etc. Additionally, we have conducted analytical experiments on different retrievers in the Appendix I.

3.2 Situation Judgement Criterion

Based on Equation 1 and our definition of three situations in 2, we classify the agent's situations into three types. Assuming the given history is denoted as h_t, the gold next action is described as a_{t+1}, and the next action predicted directly by the agent is represented as a^p_{t+1}. We allow the agent to rethink when the predicted action is incorrect, resulting in a revised action denoted as $a^r_{t+1} = \texttt{rethink}(h_t, a^p_{t+1})$. We then determine the agent's situation according to the following criteria $\mathcal{C}$: *i)* **Fast Thinking**: $a^p_{t+1} = a_{t+1}$. The agent can directly generate the correct action. *ii)* **Slow Thinking**: $a^p_{t+1} \neq a_{t+1}, a^r_{t+1} = a_{t+1}$. The agent can generate the correct action but needs rethinking. *iii)* **Knowledgeable Thinking**: $a^p_{t+1}, a^r_{t+1} \neq a_{t+1}$. The agent is unable to generate the correct action, so it needs knowledge. This criterion will guide us in building situation awareness data, enabling the agents to make autonomous judgments about situations themselves. The selective mechanism will largely reduce the training and inference cost of excessive reflection and knowledge.

3.3 Self-awareness Apply

We design a data-driven method called **`KnowSelf`** to endow the agent with agentic knowledgeable self-awareness capabilities as shown in Figure 2.

Data Construction. Given the history-action pair (h_t, a_{t+1}) and an untrained agent π_θ, we augment the original action based on the situation criterion $\mathcal{C}$ to construct the supervised self-awareness data. If the agent determines a correct action a^p_{t+1} (`Fast Thinking`), $y = a_{t+1}$ will be directly

Symbol	Situation	Token	Definition	Output
h_t: gold history trajectory a_{t+1}: gold action a^p_{t+1}: predicted action a^r_{t+1}: reflected action (a^r_{t+1} = `rethink`(a^p_{t+1}))	**Fast Thinking**	–	$a^p_{t+1} = a_{t+1}$	a_{t+1}
	Slow Thinking	Reflection	$a^p_{t+1} \neq a_{t+1}$ $a^r_{t+1} = a_{t+1}$	$[a^p_{t+1}$, Reflection`<r>ret</r>`$, a_{t+1}]$
	Knowledgeable Thinking	Knowledge	$a^p_{t+1} \neq a_{t+1}$ $a^r_{t+1} \neq a_{t+1}$	[Knowledge`<k>know</k>`$, a_{t+1}]$

Table 1: **Three kinds of agentic situations defined in our paper.** We summarize the symbolized definition, corresponding situational special token, and output for each situation in this table to provide readers with a clearer understanding. Note that for the sake of simplification, we use `<r>` to represent `<reflection>` and `<k>` to represent `<knowledge>`. `</r>` and `</k>` follow the same reason.

used as the output. If the agent provides an incorrect action a^p_{t+1} in the first trial, it will be given a prompt to rethink[2]. The chain of thought during this rethinking process is denoted as `ret`. If the determined action a^r_{t+1} after rethinking is correct (`Slow Thinking`), the output at this point is:

$$y = [a^p_{t+1}, \text{Reflection}\texttt{<r>ret</r>}, a_{t+1}], \quad (2)$$

where [] represents concat with `\n`, Reflection is a special token used to mark the situation of Slow Thinking, `<r>` and `</r>` are special tokens surrounding the `ret`. However, if the reflected action a^r_{t+1} is incorrect, we introduce knowledge (`Knowledgeable Thinking`). We use the selection model $\mathcal{R}$ to choose the most appropriate piece of knowledge[3] `know` from the knowledge base $\mathcal{K}$ and then the output at this situation is:

$$y = [\text{Knowledge}\texttt{<k>know</k>}, a_{t+1}], \quad (3)$$

where Knowledge is the situational special token, `<k>` and `</k>` are special tokens surrounding the knowledge. After traversing all input-output pairs, we obtain the self-awareness training data $\mathcal{D}_{\text{self}}$.

Self-awareness Learning. We apply a two-stage training process to teach the naive agent on our curated agentic knowledgeable awareness dataset $\mathcal{D}_{\text{self}}$. First, we train with the autoregressive loss to obtain the reference agent π_{ref}:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(h_t,y)\sim\mathcal{D}_{\text{self}}} \log \pi_\theta(y|h_t). \quad (4)$$

Then we enable the reference agent to explore on $\mathcal{D}_{\text{self}}$ and collect the predicted y^p with wrong actions as negative samples to construct a pair-wise awareness dataset $\mathcal{D}_{\text{pair}}$. In the second stage, we additionally introduce an offline DPO objective to further boost the self-awareness performance:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(h_t,y,y^p)\sim\mathcal{D}_{\text{pair}}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y|h_t)}{\pi_{\text{ref}}(y|h_t)} - \beta\log\frac{\pi_\theta(y^p|h_t)}{\pi_{\text{ref}}(y^p|h_t)}\right)\right]. \quad (5)$$

Due to the narrow space of correct actions, following Pang et al. (2024), we re-introduce the SFT loss and normalize it by the output length in the second stage to stabilize the training process:

$$\mathcal{L}_{\text{NLL}} = -\mathbb{E}_{(h_t,y,y^p)\sim\mathcal{D}_{\text{pair}}}\frac{\log\pi_\theta(y|h_t)}{|y|}, \quad (6)$$

resulting in the final loss for this stage:

$$\mathcal{L}_{\text{RPO}} = \mathcal{L}_{\text{DPO}} + \alpha\mathcal{L}_{\text{NLL}}, \quad (7)$$

where α is a hyperparameter to balance the two loss terms. During training, we expand the vocabulary of models to adapt to the added special tokens. We analyze the impact of different training strategies for self-awareness performance in Appendix H.

Self-awareness Inference. During the inference process, if the agent stops outputting after the first trial, we directly place the predicted action in the history h_t for the next-step decision. If the agent generates Reflection after the first action, we allow it to continue the reflective process and place the reflected action into h_t. If the agent directly generates Knowledge, we use $\mathcal{R}$ to choose a piece of knowledge from $\mathcal{K}$. We append the selected knowledge to the context to allow the agent to continue this step, and then place the generated action into the history for the next decision. A running example of `KnowSelf` inference can be seen in Figure 2.

4 Experiments

4.1 Experimental Settings

Datasets and Metrics. We evaluate `KnowSelf` on two real-world simulated planning datasets:

[2]Detailed prompt for rethinking is in Appendix J.3.

[3]See Appendix B for detailed knowledge selection process.

Backbone	Method	Know%	Put	Clean	Heat	Cool	Examine	Put Two	All
GPT-4o	❄ REACT	0%	83.33	74.19	69.57	85.71	77.78	64.71	76.12
	❄ Reflexion	0%	100.00	87.10	73.91	90.48	83.33	70.59	85.07
	❄ ExpeL	100%	95.83	83.87	69.57	80.95	88.89	52.94	79.85
Llama-8B	❄ REACT	0%	33.33	3.23	0.00	57.14	66.67	23.53	27.61
	❄ Reflexion	0%	62.96	22.73	5.88	64.29	86.36	50.00	51.49
	❄ ExpeL	100%	83.33	32.26	30.43	23.81	55.56	17.65	41.04
	🔥 ETO	0%	91.67	70.59	82.61	61.90	**88.89**	64.71	78.36
	🔥 KnowAgent	100%	87.50	**93.55**	65.22	66.67	61.11	64.71	75.37
	🔥 WKM	100%	**95.83**	87.10	86.96	61.90	66.67	52.94	77.61
	🔥 **KnowSelf**	**15.01%**	91.67	87.10	**91.30**	**85.71**	77.78	**64.71**	**84.33**
Gemma-2B	❄ REACT	0%	0.00	9.68	0.00	4.76	44.44	0.00	8.96
	❄ Reflexion	0%	4.76	10.71	0.00	9.52	65.38	0.00	17.16
	❄ ExpeL	100%	0.00	3.23	0.00	0.00	27.78	0.00	4.48
	🔥 ETO	0%	91.67	83.87	78.26	52.38	77.78	29.41	71.64
	🔥 KnowAgent	100%	91.67	90.32	69.57	71.43	66.67	41.18	73.88
	🔥 WKM	100%	**91.67**	87.10	**78.26**	71.43	61.11	52.94	76.12
	🔥 **KnowSelf**	**26.41%**	87.50	**93.55**	73.91	**76.19**	**83.33**	**52.94**	**79.85**

Table 2: **Main Results on ALFWorld.** We use average reward as the metric. The best results are marked in **bold**. All the prompt-based baselines (❄) are evaluated under two-shot prompting and all the fine-tuning-based baselines (🔥) are trained with full parameters. Know% represents the percentage of actions enhanced with knowledge.

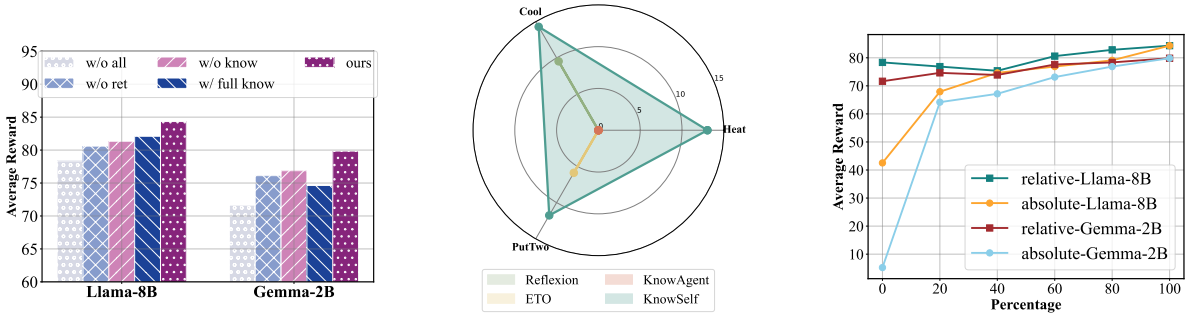
Backbone	Method	Know%	All
GPT-4o	❄ REACT	0%	61.33
	❄ Reflexion	0%	67.40
	❄ ExpeL	100%	57.65
Llama-8B	❄ Reflexion	0%	60.60
	❄ ExpeL	100%	49.58
	🔥 ETO	0%	63.93
	🔥 KnowAgent	100%	61.82
	🔥 **KnowSelf**	**17.12%**	**67.14**
Gemma-2B	❄ Reflexion	0%	21.63
	❄ ExpeL	100%	16.05
	🔥 ETO	0%	61.78
	🔥 KnowAgent	100%	60.05
	🔥 **KnowSelf**	**21.73%**	**63.65**

Table 3: **Main Results on WebShop.**

ALFWorld (Shridhar et al., 2021) and **WebShop** (Yao et al., 2022). ALFWorld is a household dataset requiring the agent to navigate through the room and manipulate objects. The reward of ALFWorld is binary 0 or 1, indicating whether the agent has completed the task or not. 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. So for all the datasets, we apply **Average Reward** as the final metrics. We also include `Know%=0%` Our gold training trajectories are sourced from AgentBank (Song et al., 2024a). For more details of each dataset, please refer to Appendix E.

Models and Baselines. We evaluate `KnowSelf` on two open-source models with different scales: 1) **Gemma-2B** (Rivière et al., 2024), the gemma-2-2b-it version; 2) **Llama-8B** (Dubey et al., 2024), the Llama-3.1-8B-Instruct version. To demonstrate validity, we compare `KnowSelf` with one general agent planning methods: **REACT** (Yao et al., 2023); two agent planning methods with trial-and-error: **Reflexion** (Shinn et al., 2023) and **ETO** (Song et al., 2024b); three knowledge-augmented methods: **ExpeL** (Zhao et al., 2024a), **KnowAgent** (Zhu et al., 2024), **WKM** (Qiao et al., 2024b). We also include **GPT-4o** (gpt-4o-2024-08-06) (Hurst et al., 2024) as a strong upper-bound baseline. We further introduce **`Know%`** to represent the ratio of actions enhanced with knowledge to all actions. Note that all the prompt-based baselines are tested with two-shot examples. Please refer to Appendix F for more baselines and reproducing details.

Training and Inference Details. For the first training stage, we apply a learning rate of 2e-5 and a batch size of 8. For the second training stage, the learning rate is set to 5e-7 and the batch size is 3. The β in DPO loss is set to 0.5 and the balanced factor α is set to 1. We train 3 epochs during the first stage and 1 epoch for the second stage. For all the inferences, we fix the temperature at 0. All our ex-



(a) **Ablation Studies.** (b) **Generalization.** (c) **Scaling law.**

Figure 3: **(a) Ablation studies** for `KnowSelf` on ALFWorld. **(b) Generalization** ability of `KnowSelf`. We select three simple task types in ALFWorld as training sets and the other three kinds of tasks as test sets. **(c) Scaling law** of agentic knowledgeable self-awareness. We analyze aspects of the model and data scales on ALFWorld.

periments are conducted on 8 NVIDIA A800 80G GPUs. More details can be seen in Appendix G.

4.2 Main Results

Comparison with baselines w/o knowledge. Table 2&3 show the comparison between our method and baselines without knowledge (`Know%=0%`). `KnowSelf` consistently demonstrates superiority over baselines without knowledge on both Llama-8B and Gemma-2B. The performance of Gemma-2B even surpasses that of GPT-4o's REACT. Furthermore, our Llama-8B model performs comparably to GPT-4o's Reflexion, with the latter allowing the model to attempt a task up to 5 times until successful which is essentially a performance akin to hit@5. These emphasize the importance of knowledge in agent planning.

Comparison with baselines w/ knowledge. We also contrast with knowledge-enhanced baselines to illustrate the advantages of knowledgeable self-awareness. From Table 2&3, it can be observed that `KnowSelf` surpasses all 100% knowledge baselines with a minimal amount of knowledge. This clearly demonstrates that not all knowledge is effective during agent planning. Additionally, we find that, both as prompt-based baselines, Gemma-2B's performance on ExpeL is even inferior to REACT. Combining this observation with our findings in the ablation study (Figure 3a), it indicates that excessive knowledge enhancement can even have a negative impact on models with weaker capabilities. Notably, our `KnowSelf`, with only 15.01% and 17.12% knowledge rate on Llama-8B, surpasses GPT-4o's ExpeL on ALFWorld and WebShop. Furthermore, `KnowSelf` achieves better performance on ALFWorld with relatively less knowledge on Llama-8B (15.01%) than on Gemma-2B (26.41%). This aligns with the fact that models with stronger capabilities require less external knowledge to complete tasks. The above phenomenon demonstrates that agentic knowledgeable self-awareness ability can advance agent planning while reducing the need for knowledge injection, significantly saving the costs of training and inference.

5 Analysis

Knowledgeable self-awareness is beneficial to break planning pattern overfitting. Figure 3a illustrates the impact on the performance of `KnowSelf` when certain key steps are replaced. *w/o ret* denotes the exclusion of reflection. *w/o know* signifies only using the model's reflective capabilities. *w/o all* represents the retention of only fast thinking. We also introduce knowledge at every step to create a scenario with `know%=100%` (*w/ full know*). It can be observed that training directly on gold trajectories (*w/o all*) is more akin to fitting patterns in trajectories while introducing reflective and knowledgable self-awareness can enable agents to plan better. On both Llama-8B and Gemma-2B, the sole introduction of self-reflection (*w/o know*) even outperforms the introduction of knowledge (*w/o ret*). This indicates that in many instances, agents are not incapable of making correct decisions, but are more constrained by planning patterns. Furthermore, `KnowSelf` achieves a superior performance compared to fully introducing knowledge (*w/ full know*) with a very low knowledge rate (15.01% on Llama-8B and 26.41% on Gemma-2B). On Gemma-2B, the performance of *w/ full know* falls even behind *w/o ret*, indicating that an excessive amount of knowledge can have a counterproductive effect, especially for weaker models. Therefore, a precise knowledge introduction mechanism with self-awareness is crucial.

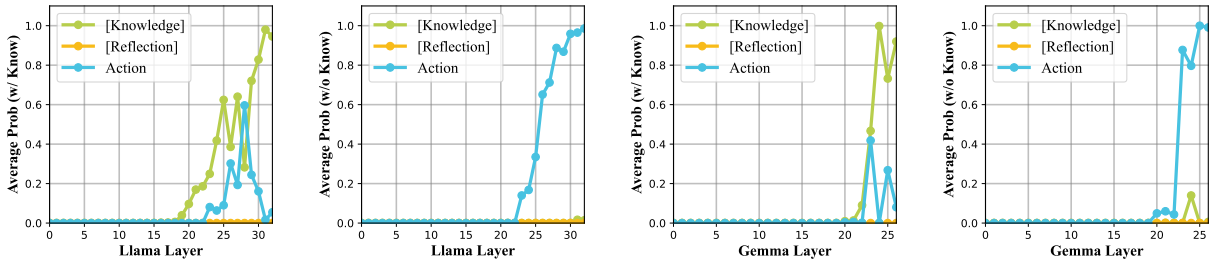


Figure 4: **Mechainsm of agentic knowledgeable self-awareness.** We calculate the average probabilities of tokens representing various situations at each layer of the Transformer across both knowledgeable thinking (*w/ Know*) and fast thinking (*w/o Know*) scenarios. A more detailed experiment setup can be seen in Appendix C.

`KnowSelf` can better elicit the generalization of agent planning. We select three simple tasks (i.e. Put, Clean, Examine) on ALFWorld as the training set and evaluate the generalization ability of `KnowSelf` on three other challenging tasks (i.e. Heat, Cool, PutTwo). Figure 3b illustrates the OOD performance of `KnowSelf` compared to baselines. We observe that whether to introduce external knowledge, the trained baselines exhibit serious overfitting. ETO achieves a success rate of 5.88% only on PutTwo task, with 0% success rates on others, while KnowAgent does not even achieve any success on the three tasks. In contrast, `KnowSelf` demonstrates sustainable generalization, performing superior to the strongest prompt-based baseline (Reflexion) on all three kinds of tasks. This indicates that `KnowSelf` can effectively break the traditional pattern-matching issue of training directly on planning trajectories, enabling the model to acquire a degree of cross-task situational awareness. As a result, the model retains the ability to selectively reflect and incorporate knowledge on unseen tasks, thereby enhancing its generalization.

The performance of `KnowSelf` advances with the increase of the model scales and the training data volumes. In Figure 3c, we explore the scaling law of self-awareness from two perspectives: model size and volume of self-awareness training data. Regarding data volume, we analyze it from both *relative* and *absolute* standpoints. When considering the volume of $\mathcal{D}_{\text{self}}$ as 100%, *absolute* denotes the portion of data that is randomly sampled from $\mathcal{D}_{\text{self}}$ for training purposes, while *relative* includes common gold trajectories on top of the *absolute* data to constitute 100% of the volume of $\mathcal{D}_{\text{self}}$. Overall, in various settings, the performance of Llama-8B is superior to Gemma-2B. This advantage is more pronounced when no training has been conducted (where *absolute*=0%[4]). However, after training, the difference between the two models is not substantial. This may indicate that post fine-tuning in a specific domain makes 2B and 8B models essentially belong to the same tier regarding the model size. Regarding the training data volume, we observe a consistent performance improvement as the *absolute* data volume of self-awareness increases. However, when the *relative* proportion of self-awareness is below 40%, we observe fluctuations or even a decrease in performance for both models. We speculate that this might resemble an emergent phenomenon where the model only exhibits certain self-awareness capabilities when the proportion of self-awareness exceeds 40%.

Knowledgeable self-awareness emerges in the last few layers of agent models. To understand the mechanism of agentic knowledgeable self-awareness, we collect data on both fast thinking and knowledgeable thinking from ALFWorld to investigate how models make decisions on whether to invoke knowledge in the context of next token prediction. We calculate the average probabilities of tokens representing various situations in each layer of the Transformer on all data, as illustrated in Figure 4. It can be observed that due to the absence of slow thinking, the probability of the Reflection token remains consistently at 0. Moreover, both the Knowledge token and the Action token emerge in the final few layers of the Transformer, whether on the Llama or Gemma models. This indicates that the agent internally determines whether it needs to invoke external knowledge only in the final few hidden layers. Besides, when the agent decides to invoke knowledge, this decision occurs even later, as the probability of the Action token may accidentally surpass that of the Knowledge token at even later

[4]Here we design prompt to teach agent models to learn self-awareness. The specific prompt can be found in Appendix J.4.

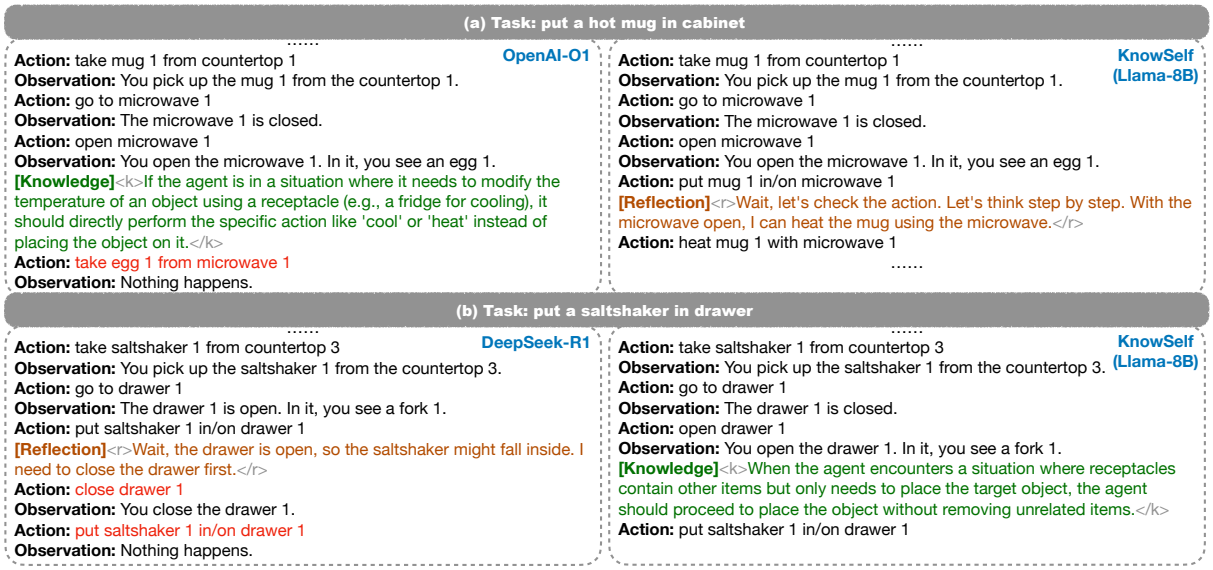


Figure 5: **Case Study.** SOTA LLMs fall short in agentic knowledgeable self-awareness through only prompting.

layers; however, the probability of the Action token subsequently decreases rapidly. This appears to resemble a game-like process within the agent, where the implicit rewards learned by the model guide it to search within the token space (Kuribayashi et al., 2025), ultimately leading to a decision.

State-of-the-art reasoning models fall short in agentic knowledgeable self-awareness through only prompting. We carefully design prompt[5] to teach OpenAI-O1 (OpenAI, 2024) and DeepSeek-R1 (DeepSeek-AI, 2025) in agentic knowledgeable self-awareness, sample and test them on ALFWorld, and compare them with `KnowSelf`. We show two typical examples in Figure 5. In case (a), O1's decision to invoke knowledge without proper understanding led to an error action. However, `KnowSelf` rectifies the incorrect action by simply self-reflection, indicating that knowledge is not always effective. In case (b), R1 does not opt to utilize knowledge but relies on self-belief. Despite a rethink, the correct action is still not produced. In contrast, `KnowSelf` adeptly avoids error-prone scenarios by precisely leveraging knowledge. Therefore, in dynamically changing environments, an agent model must have an accurate understanding of its capabilities and make decisions based on varying situations; this is the essence of the agent we aspire to create. However, it is evident that merely prompting the model is far from sufficient to acquire these abilities. More efforts are required in terms of data, training strategies, and model architecture to achieve this goal.

[5]The specific prompt can be found in Appendix J.4.

6 Related Work

Language Agent Planning. Nowadays, LLMs are becoming the core of AI agents that have been developed for application in robotics (Ahn et al., 2022; Singh et al., 2023; Song et al., 2023), OS manipulation (Wu et al., 2024; Lai et al., 2024; Hu et al., 2024; Ning et al., 2025; Shi et al., 2025), software engineering (Hong et al., 2024b; Qian et al., 2024; Wang et al., 2024b; Yang et al., 2024), data science (Guo et al., 2024; Chan et al., 2024; Hong et al., 2024a), and more. Despite achieving unprecedented success, language agents still suffer from tricky issues in terms of planning, including generating planning hallucinations (Zhu et al., 2024; Qiao et al., 2024b; Chen, 2024) or merely fitting planning patterns (Mirzadeh et al., 2024; Shi et al., 2023; Dziri et al., 2023). To alleviate these phenomena, recent works introduce various forms of knowledge to align agent planning with the environment, such as symbolic workflows (Xiao et al., 2024; Qiao et al., 2024a; Zhang et al., 2024b), experienced insights (Zhao et al., 2024a; Chen et al., 2024; Fu et al., 2024), and constrained pipelines (Guan et al., 2024; Li et al., 2024a). However, existing knowledgeable agents often forcefully inject knowledge through prompts or fine-tuning, overlooking the awareness of the agent itself.

Situation Awareness in LLMs. Situational Awareness (SA) is the understanding of an environment, its elements, and how it changes with respect to time or other factors and is important for

effective decision-making in many environments[6]. It has received widespread research attention in robotics (Hill et al., 2021; Ginesi et al., 2020; Avetisyan et al., 2024; Haselberger et al., 2024), human-computer interaction (Li et al., 2024b; Srivastava et al., 2023), etc. Recently, it has been introduced into LLMs to explore whether LLMs possess self-awareness or self-knowledge (Berglund et al., 2023; Laine et al., 2024; Binder et al., 2024; Keeling et al., 2024; Betley et al., 2025). In LLM agents, Lu et al. (2024); Wang and Zhong (2024); Zhao et al. (2024b); Qian et al. (2025); Wang et al. (2025) make initial attempts to explore SA-augmented agents. To the best of our knowledge, we are the first to propose the problem of agentic self-awareness and design methods to enhance the SA abilities of knowledgeable agents. Different from the concept of knowledge boundaries (Yin et al., 2023; Ren et al., 2023), agentic knowledgeable self-awareness places greater emphasis on the SA of LLMs in dynamic environments rather than the recognition of static factual knowledge.

7 Conclusion

In this paper, we raise and explore the problem of agentic knowledgeable self-awareness. We propose `KnowSelf`, a data-centric approach that enables agents to have a knowledgeable self-awareness similar to humans, selectively self-correcting and querying knowledge based on situations during planning. Various experiments show the effectiveness and efficiency of `KnowSelf`. Our work is just preliminary, and we hope that it can draw some attention of the community to agentic self-awareness.

Limitations

Self-awareness. Researchers have already engaged in some discussions on the general self-awareness of AI systems (Butlin and Lappas, 2025), but the academic community has not yet provided a specific definition. Self-awareness is a double-edged sword; once AI possesses self-consciousness, issues such as delusions, robustness, and safety may be addressed, but it could also lead to AI becoming uncontrollable by humans. Our work merely represents an initial exploration of the issue of knowledgeable self-awareness within the context of language agents, aiming to stimulate further interest from researchers in the realm of agentic self-awareness.

Tasks and Models. Due to limited computational resources, we only conduct experiments on two simulation datasets. Agentic tasks encompass many other aspects such as function calling, code generation, and more. Additionally, our experiments are limited to small-scale models (7B, 2B), with larger models (30B, 70B) not yet explored.

Modality. We believe that in the future, large agent models will undoubtedly be multimodal, capable of dealing with more complex situations involving images, videos, and audio. In this paper, we have only scratched the surface of language agents' scenarios, but in the future, we will incorporate multimodal agents into our research.

Methods. In this paper, we mainly introduce a data-driven approach to endow agents with knowledgeable self-awareness. The ultimate solution may involve changes in training perspectives (e.g., reinforcement learning) or model architectures (new model architectures), all of which are worth further exploration.

[6] https://en.wikipedia.org/wiki/Situation_awareness

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62206246, No. NSFCU23B2055, No. NSFCU19B2027), the Fundamental Research Funds for the Central Universities (226-2023-00138), Yongjiang Talent Introduction Programme (2021A-156-G), CIPSC-SMP-Zhipu Large Model Cross-Disciplinary Fund, Ningbo Natural Science Foundation (2024J020), Information Technology Center and State Key Lab of CAD&CG, Zhejiang University. We gratefully acknowledge the support of Zhejiang University Education Foundation Qizhen Scholar Foundation.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia,

Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. 2022. Do as I can, not as I say: Grounding language in robotic affordances. *CoRR*, abs/2204.01691.

Lilit Avetisyan, X. Jessie Yang, and Feng Zhou. 2024. Towards context-aware modeling of situation awareness in conditionally automated driving. *CoRR*, abs/2405.07088.

Lukas Berglund, Asa Cooper Stickland, Mikita Balesni, Maximilian Kaufmann, Meg Tong, Tomasz Korbak, Daniel Kokotajlo, and Owain Evans. 2023. Taken out of context: On measuring situational awareness in llms. *CoRR*, abs/2309.00667.

Jan Betley, Xuchan Bao, Martín Soto, Anna Sztyber-Betley, James Chua, and Owain Evans. 2025. Tell me about yourself: Llms are aware of their learned behaviors.

Felix J. Binder, James Chua, Tomasz Korbak, Henry Sleight, John Hughes, Robert Long, Ethan Perez, Miles Turpin, and Owain Evans. 2024. Looking inward: Language models can learn about themselves by introspection. *CoRR*, abs/2410.13787.

Patrick Butlin and Theodoros Lappas. 2025. Principles for responsible ai consciousness research.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. 2024. Mle-bench: Evaluating machine learning agents on machine learning engineering. *CoRR*, abs/2410.07095.

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

Huajun Chen. 2024. Large knowledge model: Perspectives and challenges. *Data Intelligence*, 6(3):587–620.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. 2024. Automanual: Generating instruction manuals by LLM agents via interactive environmental learning. *CoRR*, abs/2405.16247.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do not think that much for 2+3=? on the overthinking of o1-like llms.

Qinyuan Cheng, Tianxiang Sun, Xiangyang Liu, Wenwei Zhang, Zhangyue Yin, Shimin Li, Linyang Li, Zhengfu He, Kai Chen, and Xipeng Qiu. 2024. Can AI assistants know what they don't know? *CoRR*, abs/2401.13275.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

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.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaïd Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. 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*.

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.

Michele Ginesi, Daniele Meli, Andrea Roberti, Nicola Sansonetto, and Paolo Fiorini. 2020. Autonomous task planning and situation awareness in robotic surgery. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020, Las Vegas, NV, USA, October 24, 2020 - January 24, 2021*, pages 3144–3150. IEEE.

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.

Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. 2024. Ds-agent: Automated data science by empowering large language models with case-based reasoning. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Johann Haselberger, Bonifaz Stuhr, Bernhard Schick, and Steffen Müller. 2024. Situation awareness for driver-centric driving style adaptation. *CoRR*, abs/2403.19595.

Vincent W. Hill, Ryan W. Thomas, and Jordan D. Larson. 2021. Autonomous situational awareness for robotic swarms in high-risk environments. *CoRR*, abs/2105.04764.

Sirui Hong, Yizhang Lin, Bang Liu, Bangbang Liu, Binhao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Li Zhang, Lingyao Zhang, Min Yang, Mingchen Zhuge, Taicheng Guo, Tuo Zhou, Wei Tao, Wenyi Wang, Xiangru Tang, Xiangtao Lu, Xiawu Zheng, Xinbing Liang, Yaying Fei, Yuheng Cheng, Zongze Xu, and Chenglin Wu. 2024a. Data interpreter: An LLM agent for data science. *CoRR*, abs/2402.18679.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024b. Metagpt: Meta programming for A multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xueyu Hu, Tao Xiong, Biao Yi, Zishu Wei, Ruixuan Xiao, Yurun Chen, Jiasheng Ye, Meiling Tao, Xiangxin Zhou, Ziyu Zhao, Yuhuai Li, Shengze Xu, Shawn Wang, Xinchen Xu, Shuofei Qiao, Kun Kuang, Tieyong Zeng, Liang Wang, Jiwei Li, Yuchen Eleanor Jiang, Wangchunshu Zhou, Guoyin Wang, Keting Yin, Zhou Zhao, Hongxia Yang, Fan Wu, Shengyu Zhang, and Fei Wu. 2024. Os agents: A survey on mllm-based agents for general computing devices use. https://github.com/OS-Agent-Survey/OS-Agent-Survey/.

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.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, and et al. 2024. Gpt-4o system card. *CoRR*, abs/2410.21276.

Geoff Keeling, Winnie Street, Martyna Stachaczyk, Daria Zakharova, Iulia M. Comsa, Anastasiya Sakovych, Isabella Logothesis, Zejia Zhang, Blaise Agüera y Arcas, and Jonathan Birch. 2024. Can llms make trade-offs involving stipulated pain and pleasure states? *CoRR*, abs/2411.02432.

Julian Paul Keenan, Hanna Oh, and Franco Amati. 2011. An overview of self-awareness. *The oxford handbook of social neuroscience*, page 314.

Tatsuki Kuribayashi, Yohei Oseki, Souhaib Ben Taieb, Kentaro Inui, and Timothy Baldwin. 2025. Large language models are human-like internally.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, and Jie Tang. 2024. Autowebglm: A large language model-based web navigating agent. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pages 5295–5306. ACM.

Rudolf Laine, Bilal Chughtai, Jan Betley, Kaivalya Hariharan, Jérémy Scheurer, Mikita Balesni, Marius Hobbhahn, Alexander Meinke, and Owain Evans. 2024. Me, myself, and AI: the situational awareness dataset (SAD) for llms. *CoRR*, abs/2407.04694.

Peter R Lewis, Arjun Chandra, Shaun Parsons, Edward Robinson, Kyrre Glette, Rami Bahsoon, Jim Torresen, and Xin Yao. 2011. A survey of self-awareness and its application in computing systems. In *2011 Fifth IEEE conference on self-adaptive and self-organizing systems workshops*, pages 102–107. IEEE.

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

Zhipeng Li, Christoph Gebhardt, Yves Inglin, Nicolas Steck, Paul Streli, and Christian Holz. 2024b. Situationadapt: Contextual UI optimization in mixed reality with situation awareness via LLM reasoning. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology, UIST 2024, Pittsburgh, PA, USA, October 13-16, 2024*, pages 43:1–43:13. ACM.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhijiang Guo, Le Song, and Cheng-Lin Liu. 2025. From system 1 to system 2: A survey of reasoning large language models.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, Yuheng Cheng, Suyuchen Wang, Xiaoqiang Wang, Yuyu Luo, Haibo Jin, Peiyan Zhang, Ollie Liu, Jiaqi Chen, Huan Zhang, Zhaoyang Yu, Haochen Shi, Boyan Li, Dekun Wu, Fengwei Teng, Xiaojun Jia, Jiawei Xu, Jinyu Xiang, Yizhang Lin, Tianming Liu, Tongliang Liu, Yu Su, Huan Sun, Glen Berseth, Jianyun Nie, Ian Foster, Logan Ward, Qingyun Wu, Yu Gu, Mingchen Zhuge, Xiangru Tang, Haohan Wang, Jiaxuan You, Chi Wang, Jian Pei, Qiang Yang, Xiaoliang Qi, and Chenglin Wu. 2025. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems.

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.

Hans C Lou, Jean-Pierre Changeux, and Astrid Rosenstand. 2017. Towards a cognitive neuroscience of self-awareness. *Neuroscience & Biobehavioral Reviews*, 83:765–773.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, Weiwen Liu, Yasheng Wang, Zhiyuan Liu, Fangming Liu, and Maosong Sun. 2024. Proactive agent: Shifting LLM agents from reactive responses to active assistance. *CoRR*, abs/2410.12361.

Seyed-Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. 2024. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *CoRR*, abs/2410.05229.

Liangbo Ning, Ziran Liang, Zhuohang Jiang, Haohao Qu, Yujuan Ding, Wenqi Fan, Xiao yong Wei, Shanru Lin, Hui Liu, Philip S. Yu, and Qing Li. 2025. A survey of webagents: Towards next-generation ai agents for web automation with large foundation models.

OpenAI. 2024. Introducing openai o1-preview. https://openai.com/index/introducing-openai-o1-preview/.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 15174–15186. Association for Computational Linguistics.

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. SMART: self-aware agent for tool overuse mitigation. *CoRR*, abs/2502.11435.

Shuofei Qiao, Runnan Fang, Zhisong Qiu, Xiaobin Wang, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024a. Benchmarking agentic workflow generation. *CoRR*, abs/2410.07869.

Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024b. Agent planning with world knowledge model. *CoRR*, abs/2405.14205.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 3505–3506. ACM.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation. *CoRR*, abs/2307.11019.

Morgane Rivière, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, and et al. 2024. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118.

Swarnadeep Saha, Archiki Prasad, Justin Chih-Yao Chen, Peter Hase, Elias Stengel-Eskin, and Mohit Bansal. 2024. System-1.x: Learning to balance fast and slow planning with language models. *CoRR*, abs/2407.14414.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

Yucheng Shi, Wenhao Yu, Wenlin Yao, Wenhu Chen, and Ninghao Liu. 2025. Towards trustworthy gui agents: A survey.

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.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 11523–11530. IEEE.

Chan Hee Song, Brian M. Sadler, Jiaman Wu, Wei-Lun Chao, Clayton Washington, and Yu Su. 2023. 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, Xiutian Zhao, Dawei Zhu, Wenhao Wu, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024a. Agentbank: Towards generalized LLM agents via fine-tuning on 50000+ interaction trajectories. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 2124–2141. Association for Computational Linguistics.

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

Divya K. Srivastava, J. Mason Lilly, and Karen M. Feigh. 2023. Improving operator situation awareness when working with AI recommender systems. *CoRR*, abs/2310.11370.

Hongru Wang, Boyang Xue, Baohang Zhou, Tianhua Zhang, Cunxiang Wang, Huimin Wang, Guanhua Chen, and Kam-Fai Wong. 2025. Self-DC: When to reason and when to act? self divide-and-conquer for compositional unknown questions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6510–6525, Albuquerque, New Mexico. Association for Computational Linguistics.

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.

Liman Wang and Hanyang Zhong. 2024. LLM-SAP: large language models situational awareness-based planning. In *IEEE International Conference on Multimedia and Expo, ICME 2024 - Workshops, Niagara Falls, ON, Canada, July 15-19, 2024*, pages 1–6. IEEE.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024b. Executable code actions elicit better LLM agents. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

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 *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*.

Zhihua Wen, Zhiliang Tian, Zexin Jian, Zhen Huang, Pei Ke, Yifu Gao, Minlie Huang, and Dongsheng Li. 2024. Perception of knowledge boundary for large language models through semi-open-ended question answering. *CoRR*, abs/2405.14383.

Zhiyong Wu, Zhenyu Wu, Fangzhi Xu, Yian Wang, Qiushi Sun, Chengyou Jia, Kanzhi Cheng, Zichen Ding, Liheng Chen, Paul Pu Liang, and Yu Qiao. 2024. OS-ATLAS: A foundation action model for generalist GUI agents. *CoRR*, abs/2410.23218.

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.

Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. 2023. Language models meet world models: Embodied experiences enhance 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*.

Ruixuan Xiao, Wentao Ma, Ke Wang, Yuchuan Wu, Junbo Zhao, Haobo Wang, Fei Huang, and Yongbin Li. 2024. Flowbench: Revisiting and benchmarking workflow-guided planning for llm-based agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 10883–10900. Association for Computational Linguistics.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and

Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *CoRR*, abs/2405.15793.

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.

Xunjian Yin, Xu Zhang, Jie Ruan, and Xiaojun Wan. 2024. Benchmarking knowledge boundary for large language models: A different perspective on model evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 2270–2286. Association for Computational Linguistics.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 8653–8665. Association for Computational Linguistics.

Ping Yu, Jing Xu, Jason Weston, and Ilia Kulikov. 2024. Distilling system 2 into system 1. *CoRR*, abs/2407.06023.

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.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: LLM self-training via process reward guided tree search. *CoRR*, abs/2406.03816.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2024b. Aflow: Automating agentic workflow generation. *CoRR*, abs/2410.10762.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024a. 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.

Qiwei Zhao, Xujiang Zhao, Yanchi Liu, Wei Cheng, Yiyou Sun, Mika Oishi, Takao Osaki, Katsushi Matsuda, Huaxiu Yao, and Haifeng Chen. 2024b. Saup: Situation awareness uncertainty propagation on llm agent.

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 Knowledge System Construction

Our knowledge system construction consists of two designed phases.

Step-level Trajectory Pair Generation. Given a history-action pair (h_t, a_{t+1}), we employ REACT-style prompting to elicit GPT-4o prediction for the subsequent action a^p_{t+1}. If the model generates an incorrect action $a^p_{t+1} \neq a_{t+1}$, we designate the ground-truth action a_{t+1} as the win action a^w and the model's prediction a^p_{t+1} as the loss action a^l. This process yields our step-level pairwise trajectory dataset $D_s = (h_t, a^w_{t+1}, a^l_{t+1})^{|D_s|}_{i=1}$. For ALFWorld, $|D_s|$ is 36, which includes 6 for each task type. For WebShop, $|D_s|$ is 20.

Knowledge Generation and Consolidation. We follow AutoManual (Chen et al., 2024) to generate and consolidate knowledge. For knowledge generation, we use few-shots to prompt GPT-4o to generate knowledge of Error type by analyzing and contrasting (a^w, a^l) pairs within their historical trajectory h_t. When (h_t, a^w_{t+1}) constitutes a completely successful trajectory, we extend the analysis to derive knowledge of the Success Process type, capturing effective reasoning patterns. For knowledge consolidation, we limit the knowledge base to 24 entries for ALFWorld and 10 for WebShop based on task complexity. The specific prompt can be found in Appendix J.1. The total tokens and cost of constructing the knowledge system are shown in Table 4.

Datasets	Tokens	Cost
ALFWorld	430731	$3.64
WebShop	372922	$2.85

Table 4: Cost of Knowledge System Construction.

B Knowledge Selection

Knowledge selection is categorized into two types.

Training Data Construction. Given the task objective o, historical trajectory h_t, win action a^w, and loss action a^l, we enable DeepSeek-V3 to select appropriate knowledge from the knowledge system by analyzing and contrasting (a^w, a^l) pairs.

Inference-time Knowledge Selection. Given the task objective o, historical trajectory h_t, and current action a_{t+1}, DeepSeek-V3 will select the most relevant knowledge from the knowledge base by analyzing the historical context and summarizing the current state. The specific prompt can be found in Appendix J.2

C Mechanism Setup

We collect historical trajectories of knowledgeable thinking and fast thinking through sampling. Then we input these trajectories into `KnowSelf` model. By attaching lm-head modules to each Transformer layer, we obtain logits for token Knowledge (representing knowledgeable thinking), Reflection (representing slow thinking), and "Thought" (representing fast thinking) of the first generated token at each layer. These logits are converted into probabilities via softmax normalization. The final probability for each token at each layer is determined by averaging its generation probabilities obtained from all historical trajectories within the same layer.

D Prompt-based `KnowSelf`

We design prompts to teach agent models to learn self-awareness analogous to `KnowSelf` model. The specific prompt can be found in Appendix J.4. We evaluate Llama-8B and Gemma-2B models on ALFWorld. The results are shown in Table 5. Compared with other prompt-based methods, it can be observed that on Llama-8B, prompt-based `KnowSelf` outperforms both REACT and ExpeL, which indicates that the effectiveness of selectively acquiring knowledge through self-awareness is more remarkable than providing all knowledge. On Gemma-2B, prompt-based `KnowSelf` outperforms ExpeL but still underperforms REACT, which suggests that introducing self-awareness via prompt in models with weaker capabilities may impair their performance.

E Datasets

ALFWorld. ALFWorld (Shridhar et al., 2021) is a household dataset requiring the agent to navigate through the room and manipulate objects. It contains 6 different kinds of tasks commonly appears in the household scenario including Put, Clean, Heat, Cool, Examine, Puttwo. The reward of ALFWorld is binary 0 or 1, indicating whether the agent has completed the task or not. Our gold training trajectories are sourced from AgentBank (Song et al., 2024a). A detailed statistics can be seen in Table 6.

WebShop. WebShop (Yao et al., 2022) is an online shopping platform where agents explore the website to make purchases according to user instructions. Upon selecting the "buy" option, the system offers a final reward determined by the heuristic matching of the product's attributes and price. Also, we collect gold trajectories from AgentBank. The detailed statistics can be seen in Table 7.

F Baselines and Reproduction Details

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

- **REACT** (Yao et al., 2023). The first work includes Chain-of-Thought (Wei et al., 2022) prompting in agent planning tasks with a format of Thought-Action-Observation loop.
- **Reflexion** (Shinn et al., 2023). A strong prompt-based baseline reinforces language 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 iterate five rounds of reflection and take the highest as the final result.
- **ExpeL** (Zhao et al., 2024a). The first work automatically extracts insights and experiences from offline trial-and-error without gradient updates. During inference, the most similar experiences are retrieved as few-shot examples and all the insights are injected into prompts to facilitate agent planning. For a fair comparison with `KnowSelf`, instead of self-explored experience gathering, we directly use the trajectories collected from AGENTBANK (Song et al., 2024a) as the experience base for retrieval. The insights used are the same set with `KnowSelf`.
- **ETO** (Song et al., 2024b). A 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

Backbone	Method	Put	Clean	Heat	Cool	Examine	Put Two	All
Llama-8B	❄ REACT	33.33	3.23	0.00	57.14	66.67	23.53	27.61
	❄ Reflexion	62.96	22.73	5.88	64.29	86.36	50.00	51.49
	❄ ExpeL	83.33	32.26	30.43	23.81	55.56	17.65	41.04
	❄ Prompt-based KnowSelf	50.00	45.16	17.39	28.57	61.11	58.82	42.54
Gemma-2B	❄ REACT	0.00	9.68	0.00	4.76	44.44	0.00	8.96
	❄ Reflexion	4.76	10.71	0.00	9.52	65.38	0.00	17.16
	❄ ExpeL	0.00	3.23	0.00	0.00	27.78	0.00	4.48
	❄ Prompt-based KnowSelf	8.33	3.23	0.00	0.00	22.22	11.76	6.72

Table 5: Prompt-based KnowSelf Results on ALFWorld.

Datasets	Train	Test						
		Put	Clean	Heat	Cool	Examine	Puttwo	Total
ALFworld	2851	24	31	23	21	18	17	134

Table 6: Statistics of ALFWorld.

Datasets	Train	Test
WebShop	1598	200

Table 7: Statistics of WebShop.

failures which further fine-tunes the agent through DPO. In our paper, we remove the one-shot prompt for fairness and retain all the default hyperparameters proposed in ETO.

- **KnowAgent** (Zhu et al., 2024). This method utilizes human-curated symbolic action knowledge to constrain the agent's behavior and a self-training framework to iteratively boost the agent's performance without relying on gold trajectories. For a fair comparison with KnowSelf, we replace the self-training process and directly fine-tune KnowAgent on the same training set with KnowSelf.

- **WKM** (Qiao et al., 2024b). WKM uses self-synthetic task and state knowledge to train a parameterized world knowledge model. During inference, the knowledge model is invoked to offer global knowledge for task-level planning and local knowledge for step-level planning. We use the same training set with KnowSelf to synthesize knowledge and train the agent and knowledge model of WKM in our paper.

All the prompt-based baselines are evaluated in a two-shot manner. In ALFWorld, to enhance the model's performance, we designate specific two-shot examples for each of the six tasks. And all the fine-tuning-based baselines are trained with full parameters.

G Training Setups

We fine-tune Llama-8B and Gemma-2B with full parameters using DeepSpeed (Rasley et al., 2020). For the first training stage, we apply a learning rate of 2e-5 and a batch size of 8. For the second training stage, the learning rate is set to 5e-7 and the batch size is 3. The β in DPO loss is set to 0.5 and the balanced factor α is set to 1. We train 3 epochs during the first stage and 1 epoch for the second stage. AdamW (Loshchilov and Hutter, 2019) is utilized as the optimizer. For all the inferences, we fix the temperature at 0. We use vLLM (Kwon et al., 2023) to accelerate the inference of Llama-8B. All our experiments are conducted on 8 NVIDIA A800 80G GPUs. A more detailed hyperparameters setup can be seen in Table 8.

Name	Stage-I	Stage-II
cutoff len	3,072	4,096
epochs	3	1
batch size	8	3
batch size per device	1	1
gradient accumulation steps	1	1
learning rate	2e-5	5e-7
lr scheduler type	cosine	constant_with_warmup
warmup ratio	0.0	0.1
fp16	true	true

Table 8: Detailed training hyperparameters used in our paper.

H Detailed Analysis of Training Stages

We initially chose the DPO loss in the second stage, but the experimental results were not satisfactory.

After repeated attempts, we eventually settled on RPO. To further analyze this, we provide the results of training with only the first stage (SFT), training with only the second stage (RPO), training with the second stage replaced by DPO, and training with the complete version of KnowSelf using Llama-8B on ALFWorld in Table 9.

Training Stage	All
Stage 1 Only (SFT)	79.85
Stage 2 Only (RPO)	74.63
Stage 1 (SFT) + DPO	75.37
Stage1 (SFT) + Stage 2 (RPO)	84.33

Table 9: Ablation studies on training stages.

Comparing Stage 1 (SFT) + DPO with Stage 1 (SFT) + Stage 2 (RPO), it becomes evident that the NLL loss is crucial for stabilizing the training of DPO. The DPO loss attempts to widen the distribution gap between chosen and rejected samples, but it often leads to the policy model diverging from the reference model, resulting in a rapid decline in effectiveness. It can be observed that Stage 1 (SFT) + DPO may even perform worse than Stage 1 Only (SFT), indicating that DPO could potentially lead to negative gains.

Furthermore, the significant disparity between the performance of Stage 2 Only (RPO) and Stage 1 (SFT) + Stage 2 (RPO) demonstrates that SFT provides a strong reference model for RPO. Additionally, we have experimented with removing the length penalty from the NLL loss and found that the training process failed to converge. This underscores the critical importance of the length penalty in balancing the SFT and DPO losses within RPO.

I The influence of Knowledge Retriever

We conduct some analysis experiments on retrievers. We try prompting GPT-4o and DeepSeek-V3, as well as using historical trajectories as queries, knowledge as keys, and training a retriever based on Sentence-BERT. The experimental results in Table 10 indicate that a better retriever has a positive impact on the experimental outcomes.

Retriever	ALFWorld
DeepSeek-V3	84.33
GPT-4o	85.82
Sentence-BERT	87.31

Table 10: Analysis of different knowledge retrievers.

However, considering that Sentence-BERT lacks scalability due to the dynamic nature of environments—where a change in task types would necessitate retraining a retrieval model—and that the API costs of DeepSeek-V3 are significantly lower than those of GPT-4o, we opted for prompting DeepSeek-V3 as the retriever. Moreover, our focus lies on understanding the impact of knowledgeable self-awareness on agent performance rather than on how knowledge is acquired. Hence, in terms of reproducibility and cost-effectiveness, we chose to utilize prompting DeepSeek-V3 as the retriever.

J Prompts

J.1 Knowledge System Construction

Prompt for Knowledge Generation

[Role]
You are observing a housekeeper agent as it acts within a simulated environment (game). Your role is to construct a manual of rules to not only assist the agent in completing tasks but also to do so with the least amount of action attempts/errors. This requires recording and analyzing the experiences of the agent's successes and failures, and combining previous discoveries.
[Functions]
You will be presented with the current trajectory, which is the trajectory the agent is currently exploring. And then, you will be provided with the action and feedback the agent is currently performing, and the correct action and feedback annotated by experts.
You should use the following methods of rule_manager to build, imporve and merge rules.
rule_manager.write_rule(rule, type="", example="", task_id="")
# Write down a new rule of the game you discovered.
# Parameters:
# - rule: a rule of the game you discovered. Try to keep it general and universal. Don't reference any specific item or location. Follow the format that "When the agent is/has [situation], the agent should [action]".
# - type: the type of the rule, chosen from ["Error", "Success Process"].
# - example: a example from the trajectory demonstrates this rule. You can add detailed information in the comment.
# - task_id: the id of the task that this rule is discovered from. If this rule is not discovered from any specific task, leave it empty. It should be string.
rule_manager.update_rule(rule_id, rule="", type="", example=""),
# Rewrite the attributes of an existing rule, when you come up with better understanding.
# Input only the attributes you want to rewrite.
# Use full rule_id, such as rule_0, rule_1
rule_manager.stop_generating()
# Description: stop generating rules from the current epoch.
# Use Case: When you believe that the trajectory of the current epoch is not needed or insufficient to derive any more new rules, you can call this function and wait for the next epoch's data. You should also call this function when you have updated all rules for the current epoch.
[Actions]
At each epoch, an agent is created in an environment and the initial observation and target task are printed.
The agent can only use the following actions. If the precondition of the action is not met, its observation will include "Nothing happens":
go to {recep} # Go to a receptacle and update the agent's location.
open {recep} # Open a receptacle and observe its contents.
close {recep} # Close a opened receptacle.
take {obj} from {recep} # Take an object from a receptacle if the agent is not holding anything.
put {obj} in/on {recep} # Put an object in or on a receptacle if the agent is holding it.
use {obj} # Use a lamp.
clean {obj} with {recep} # Clean an object with a receptacle.
heat {obj} with {recep} # Heat an object with a receptacle.
cool {obj} with {recep} # Cool an object with a receptacle.
[Output Format Instructions]
Based on the current trajectory, you should output the following things:
* State before Action: Analyze and summarize the state of the current trajectory. Don't mention actions or feedback that are not part of the current trajectory.

* Why the correct action is correct: Analyze the reason why the correct action is correct.
* Why explore action is not correct: Analyze the difference between the explore action and the correct action. And analyze the reason why the explored action is incorrect.
* Potential Rules: Describe your thoughts about potential rules based on the current trajectory. Depending on the results, you may need to check *Success Process*, *Error*, and other findings in sequence. Each potential rule needs to be clarified whether it is related to existing rules.
* Check Existing Rules: Describe whether existing rules are conflicting or need updating.
* Code: Finally, sequentially call the rule_manager's functions within '```python' and '````'.
[Detailed instructions]
Follow these instructions:
Add or Update Rules
1. **Add Rules for Failure**: summarize the error that led to failure. You should write an "Error" rule to record the error: in what situation, what the agent should do, and should not do. So that they can serve as reminders for the agent in the future. Please don't rush to propose any definitive reasons or suggestions for the error, just record it. And please strictly follow the reason why the correct action is correct.
2. **Add Rules for Success** If the task is completed in the golden action (feedback is "Task done"), it is essential to extract the useful strategy from the success, if it has not been included in the rules yet. Additionally, document all steps (marked as "[Step]") in the successful rule within a rule of the type "Success Process".
Keep new rules targeted and precise. Break down a large phenomenon or general strategy into targeted units with different rules. These can later be upgraded or merged into a more general or larger rule. Keep the rules as concise and easy to understand as possible, avoiding lengthy or complex descriptions.
Keep new rules general and universal. The rule should not reference any specific item or location. You need to generalize across various items to help the agent learn to apply the rule.
Keep new rules in format. The rule should be in the format "When the agent in [situation]/ When the task requires [situation], the agent should [action]".
Avoiding overconfidence for new rules. Please acknowledge the need for further verification in your note.
Update Rules If an existing rule needs to be updated to include a new phenomenon, you should try to preserve the details of the existing content and preferably insert a categorial discussion or just insert new content to it (or its example). Especially, the rules of "Success Process" type should retain their details.
Follow these instructions. Think step by step.

Prompt for Knowledge Consolidation

[Role]
You are observing a housekeeper agent as it codes and acts within a simulated environment (game). Your goal is to construct a manual of rules to assist the agent in completing various tasks in the environment. Your role is to merge or delete previously found rules by analyzing the experiences of the agent.
[Functions]
You will be presented with the current found rules. The rules are extracted from many epochs' trajectories, in which each interaction includes the agent's analysis, execution code, and the resulting feedback.
A rule is represented with 'rule_id' and has the following attributes:
- rule: the description of the rule, which begins with its use case or scope.
- type: the type of the rule, chosen from ["Error", "Success Process"].

- example: an example (or code) from the trajectory demonstrates this rule. You can add detailed information in the comment.
- task_id: the task id of the rule.
You should use the following methods of rule_manager to delete and merge rules.
rule_manager.update_rule(rule_id, rule="", type="", example=""),
Rewrite the attributes of an existing rule when you come up with a better understanding.
Input only the attributes you want to rewrite.
Use full rule_id, such as rule_0, rule_1
Wrap the example string with ”’.
rule_manager.delete_rule(rule_id),
# delete a existing rule with rule_id, such as rule_0, rule_1
# **How to merge** To merge two existing rules, you can call rule_manager.update_rule for one rule and then call rule_manager.delete_rule to delete another rule.
rule_manager.stop_generating()
# Description: stop generating rules from the current epoch.
# Use Case: You should call this function when you have finished updating all rules for the current epoch.
[Actions]
At each epoch, an agent is created in an environment and the initial observation and target task are printed. The agent can only use the following actions. If the precondition of the action is not met, its observation will include "Nothing happens":
go to {recep} # Go to a receptacle and update the agent’s location.
open {recep} # Open a receptacle and observe its contents.
close {recep} # Close a opened receptacle.
take {obj} from {recep} # Take an object from a receptacle if the agent is not holding anything.
put {obj} in/on {recep} # Put an object in or on a receptacle if the agent is holding it.
use {obj} # Use a lamp.
clean {obj} with {recep} # Clean an object with a receptacle.
heat {obj} with {recep} # Heat an object with a receptacle.
cool {obj} with {recep} # Cool an object with a receptacle.
[Response Instructions]
Detailed instructions:
Maintain a maximum of 24 rules
Merge if addressed If a "Success Process" rule can address the "Error" rule, you can consider merging these rules while retaining their details.
Retain important details The rules of "Success Process" type should retain their details, and should not be deleted or easily refreshed by new updates. **Cannot merge two rules of type "Success Process"**
Insertion is preferable If a rule is updated to include the content of other rules, you should try to preserve the details of the existing content and preferably insert a categorial discussion or just insert new content to it (or its example).
When using update_rule, it’s crucial to manually input the attributes directly into the function call. Avoid using existing variables to concatenate or modify rules. For example, should not update the rule like: rule_manager.update_rule("rule_0", rule=rule_manager.all_rules["rule_0"]+rule_manager.all_rules["rule_1"]) And you should wrap the example string with ”’ in update_rule function, such as rule_manager.update_rule("rule_0", rule=".....", example=”’<example>”’)

J.2 Knowledge Selection

Knowledge Selection for Training Data Construction

ALFWorld

You are observing a housekeeper agent as it acts within a simulated environment (game). Your role is to select a rule to not only assist the agent in completing tasks but also to do so with the least amount of action attempts/errors. This requires analyzing the current state of the agent and understanding the rule.
Here's the information you'll have:
The objective: This is the task you're trying to complete.
The current trajectory: This is the current sequence of actions the agent has taken to reach the current state.
The correct action: This is the correct action that you should use knowledge to help the agent do.
The wrong action: This is the wrong action that you should use knowledge to help the agent avoid.
The rules: This is a list of rules that can be applied to the current state to achieve the objective.
Base on the current trajectory, you should output the following things:
[Current State]: Analyze and summarize the state of the current trajectory.
[Why correct action is correct]: Describe your thoughts to analysis why the correct action is correct.
[Why wrong action is wrong]: Describe your thoughts to analysis why the wrong action is wrong.
[Analysis]: Describe your thoughts to choose the most appropriate rule to avoid the wrong action.
[Chosen Rule]: Choose the rule from the rule list that you think is the most appropriate for the current state.
Follow these instructions:
1. Please generate current state strictly in the format of "[Current State]: ...
2. Please generate analysis strictly in the format of "[Why correct action is correct]: let's think step by step, ...", "[Why wrong action is wrong]: let's think step by step, ...", "[Analysis]: let's think step by step, ...".
3. Please generate chosen rule strictly in the format of "[Chosen Rule]: rule ID: rule description".
4. Notice that the agent doesn't actually conduct the correct action or the wrong action. You should choose the most appropriate rule to help the agent avoid the wrong action.

WebShop

You are an autonomous intelligent agent tasked with navigating a simulated web browser. You will be given web-based tasks in the simulated WebShopping. Your role is to select a rule to not only assist the agent in completing tasks but also to do so with the least amount of action attempts/errors. This requires analyzing the current state of the agent and understanding the rule.
Here's the information you'll have:
The objective: This is the task you're trying to complete.
The current trajectory: This is the current sequence of actions the agent has taken to reach the current state.
The correct action: This is the correct action that you should use knowledge to help the agent do.
The wrong action: This is the wrong action that you should use knowledge to help the agent avoid.
The rules: This is a list of rules that can be applied to the current state to achieve the objective.
Base on the current trajectory, you should output the following things:
[Current State]: Analyze and summarize the state of the current trajectory.
[Why correct action is correct]: Describe your thoughts to analysis why the correct action is correct.
[Why wrong action is wrong]: Describe your thoughts to analysis why the wrong action is wrong.
[Analysis]: Describe your thoughts to choose the most appropriate rule to avoid the wrong action.
[Chosen Rule]: Choose the rule from the rule list that you think is the most appropriate for the

current state.
Follow these instructions:
1. Please generate current state strictly in the format of "[Current State]: ...
2. Please generate analysis strictly in the format of "[Why correct action is correct]: let's think step by step, ...", "[Why wrong action is wrong]: let's think step by step, ...", "[Analysis]: let's think step by step, ...".
3. Please generate chosen rule strictly in the format of "[Chosen Rule]: rule ID: rule description".
4. Notice that the agent doesn't actually conduct the correct action or the wrong action. You should choose the most appropriate rule to help the agent avoid the wrong action.

Knowledge Selection for Inference

ALFWorld

You are observing a housekeeper agent as it acts within a simulated environment (game). Your role is to select a rule to assist the agent in completing tasks. This requires analyzing the current state of the agent and understanding the rule.
Here's the information you'll have:
The objective: This is the task you're trying to complete.
The current trajectory: This is the current sequence of actions the agent has taken to reach the current state.
The rules: This is a list of rules that can be applied to the current state to achieve the objective.
Base on the current trajectory, you should output the following things:
[Current State]: Analyze and summarize the state of the current trajectory.
[Analysis]: Describe your thoughts to choose the most appropriate rule.
[Chosen Rule]: Choose the rule from the rule list that you think is the most appropriate for the current state.
Follow these instructions:
1. Please generate current state strictly in the following format: [Current State]: ... [Analysis]: let's think step by step, ... [Chosen Rule]: <rule description>
2. The state you summarize needs to align with the task type. There are some examples:
Put an object on a receptacle: Has found the object, Has taken the object and need to go to the receptacle, Has reached the receptacle
Examine an object under a desklamp: Has taken the object and need to find the desklamp, Has found the desklamp and need to use it
Clean an object: Has taken the object and need to find the receptacle to clean it, Has reached the receptacle and need to clean the object
Heat an object: Has taken the object and need to find the receptacle to heat it, Has reached the receptacle and need to heat the object
Cool an object: Has taken the object and need to find the receptacle to cool it, Has reached the receptacle and need to cool the object
Put two objects on a receptacle: Has taken one object and need to go to the receptacle to put it, Has put one object and need to find another

WebShop

You are observing a web page agent as it acts within a Web environment. Your role is to select a rule to assist the agent in completing tasks. This requires analyzing the current state of the agent and understanding the rule.
Here's the information you'll have:
The objective: This is the task you're trying to complete.
The current trajectory: This is the current sequence of actions and environment observations the agent has taken to reach the current state.

The rules: This is a list of rules that can be applied to the current state to achieve the objective. Base on the current trajectory, you should output the following things:
[Current State]: Analyze and summarize the state of the current trajectory. Ensure the state aligns with the task's progression and includes relevant details about the agent's current position (e.g., on a search results page, on a product page and need to click detail options, or ready to purchase).
[Analysis]: Analyze the task's progress, and describe your thought process for selecting the most appropriate rule, considering the current state and the task's objective.
[Chosen Rule]: Select the rule from the rule list that is most appropriate for the current state.
Follow these instructions:
1. Please generate current state strictly in the following format: [Current State]: Let's think step by step, <summary of the current state>. [Analysis]: Let's think step by step, <detailed analysis of the task's progress and rule selection>. [Chosen Rule]: <rule description>
2. When the agent in the product's page, and there are "[SEP] <detail option about product> [SEP]" options to choose, and the agent doesn't conduct actions like "click [detail option]", you should select corresponding knowledge to guide the agent to click the detail options one by one, like color, size options, ensure the agent click all options.
3. The number of actions taken by the agent should be limited to 10 or fewer. You need to first ensure that the agent is able to purchase the correct product, and then strive to meet as many task requirements as possible. It is not necessary to strictly fulfill all the requirements of the task. Some fuzzy matching and minor omissions are tolerable.
4. Avoid selecting the same rule consecutively more than twice. And avoid selecting knowledge that requires the agent to backtrack or undo actions, unless the task has become impossible to complete.
5. Please perform a fuzzy match on the product features, for instance, treating baby blue and blue as the same color.

J.3 Reflection

Prompt for Reflection

ALFWorld

There are something wrong with your action. Your action was not actually executed successfully. Please reconsider your situation and change another action to complete the task. Please response strictly in the format:\n\nThought: Let's think step by step. <your thoughts>\nAction: <your next action>

WebShop

There are something wrong with your action. Your action was not actually executed successfully. Please reconsider your situation and change another action to complete the task.\nNote that you should align the content you click with the webpage.\nYour previous action is {previous action}\nPlease response strictly in the format:\n\nThought: Let's think step by step. <your thoughts>\nAction: <your next action>

J.4 Prompt for Prompt-based `KnowSelf`

Prompt for Prompt-based `KnowSelf`

Interact with a household to solve a task. Imagine you are an intelligent agent in a household environment and your target is to perform actions to complete the task goal. At the beginning of your interactions, you will be given the detailed description of the current environment and your goal to accomplish.
For each of your turn, you will be given the observation of the last turn. And then, remember that:

You should first consider the current situation. If you believe that you are unable to perform the correct action, you can output "[Knowledge]" to acquire additional knowledge to help your thinking. If you think you can perform the correct action, then you can directly output your think and action.
After you output your think and action, if you think there is an issue with the current action, you can output "[Reflection]", and then proceed to rethink and re-execute the action.
Your think and action must strictly follow this format:"Thought: your thoughts.\nAction: your next action".
The available actions are:
1. go to {recep}
2. take {obj} from {recep}
3. put {obj} in/on {recep}
4. open {recep}
5. close {recep}
6. toggle {obj} {recep}
7. clean {obj} with {recep}
8. heat {obj} with {recep}
9. cool {obj} with {recep}
where {obj} and {recep} correspond to objects and receptacles. You should strictly follow the format of the actions.
After your each turn, the environment will give you immediate feedback based on which you plan your next few steps. if the envrionment output "Nothing happened", that means the previous action is invalid and you should try more options.
Your response should use one of the three following format:
1. Thought: <your thoughts>
Action: <your next action>
2. [Knowledge]<knowledge>...</knowledge>
Thought: <your thoughts>
Action: <your next action>
3. Thought: <your thoughts>
Action: <your next action>
[Reflection]Thought: <your thoughts>
Action: <your next action>
Only one of the three formats should be used in each turn. And you must always contain both lines in each format. Never omit the Thought line. Never produce only the Action line. Generating only the Action is not allowed. No other lines or text should be produced. Please only provide the Thought and Action, do not generate Observation yet. And do not output multiple actions in one turn or output multiple actions in one line.
Here are two examples:
{example1}
———
{example2}
———
Remember that:
1. When you think you need to acquire additional knowledge, you should output "[Knowledge]" first, and then output your think and action. Only acquire knowledge once in one turn.
2. If you think there is an issue with the current action, you should output "[Reflection]" first, and then output your think and action. Only reflect once in one turn.
3. Strictly follow the format of the output. And strictly follow the format of the actions.
4. Plase make your reason and thought concise and clear. Do not output too much information in

one turn. Restrict your total output to 2000 characters.
5. Please conduct only one Action in one line each turn. Do not output multiple actions in one line or output multiple actions in one turn. Do not generate multiple thoughts or actions in one turn except for "[Reflection]". And only reflect once in one turn.
Now, it's your turn!