SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2026, Vol. 69, Iss. 3, 132104:1–132104:15
https://doi.org/10.1007/s11432-024-4554-x

WESE: weak exploration to strong exploitation for LLM agents

Xu HUANG[1], Weiwen LIU[2], Xiaolong CHEN[1], Xingmei WANG[1], Defu LIAN[1*], Yasheng WANG[2], Ruiming TANG[2] & Enhong CHEN[1]

[1]*School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China*
[2]*Huawei Noah's Ark Lab, Shenzhen 518100, China*

Received 8 May 2024/Revised 8 January 2025/Accepted 13 August 2025/Published online 21 January 2026

Abstract Recently, large language models (LLMs) have demonstrated remarkable potential as autonomous agents. However, existing studies mainly focus on enhancing the reasoning or decision-making abilities of the agent through well-designed prompt engineering or task-specific fine-tuning, while neglecting the procedure of exploration and exploitation. When addressing complex tasks within open-world interactive environments, these methods exhibit limitations. First, the lack of global environmental information leads to greedy decisions, resulting in suboptimal solutions. By contrast, irrelevant information acquired from the environment not only adversely introduces noise but also incurs additional cost. To address these limitations, this paper proposes a novel approach, namely, weak exploration to strong exploitation (WESE), to enhance LLM agents for solving open-world interactive tasks. Specifically, WESE involves the decoupling of the exploration and exploitation process, employing a cost-effective weak agent to perform exploration tasks for global knowledge. A knowledge graph-based strategy is then introduced to store the acquired knowledge and retrieve task-relevant knowledge, enhancing the stronger agent in terms of success rate and efficiency for the exploitation task. Our approach is flexible enough to incorporate various methods in solving diverse tasks, thereby achieving remarkable improvements in effectiveness (success rates), efficiency (number of steps), and cost (expenses for API tokens) across four interactive benchmarks.

Keywords artificial intelligence, large language models, autonomous agents, exploration and exploitation, multiagent systems

Citation Huang X, Liu W W, Chen X L, et al. WESE: weak exploration to strong exploitation for LLM agents. Sci China Inf Sci, 2026, 69(3): 132104, https://doi.org/10.1007/s11432-024-4554-x

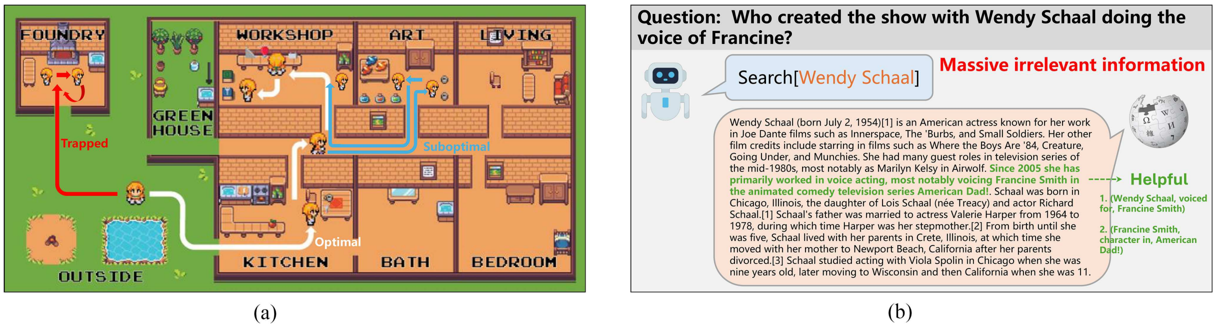
1 Introduction

Large language models (LLMs) demonstrate a myriad of capabilities across diverse domains, including human-computer conversation, instruction following, reasoning, planning, and few-shot learning [1]. These comprehensive abilities form a robust foundation that positions LLMs as intelligent agents capable of solving open-world tasks, such as household tasks and open-world question answering [2,3]. Recently, numerous studies have investigated the potential of LLM agents to enhance their capabilities in solving open-world tasks [2,4].

Benefiting from the capabilities of LLMs in instruction following and few-shot learning, most methods guide LLMs in decision-making tasks through human-crafted natural language prompts, thus avoiding the costly fine-tuning of the models [5–8]. Existing prompt-engineering approaches primarily consider two factors: incorporating task-relevant information into prompts and eliciting the reasoning capabilities of LLMs through prompt design. Task-relevant information includes task descriptions and contextual feedback, such as questions and pertinent task statements in question-answering tasks, as well as textual materials retrieved by the agent from the web during problem-solving. To enhance the reasoning capabilities of LLM agents, methods such as CoT [5], ReAct [7], and Reflexion [9] encourage LLMs to engage in explicit reasoning by constructing few-shot examples with detailed reasoning paths.

However, open-world tasks simulate real environments, wherein an agent explores and interacts continuously with its surroundings to acquire information for solving complex tasks [10–12]. These tasks possess several characteristics that make them particularly challenging. The capabilities of LLM agents remain far from optimal due to the following challenges. (1) Complexity. Each task involves multistep actions and may have multiple feasible solutions. (2) Uncertainty. The agent cannot obtain all necessary information from the initial task description and must

* Corresponding author (email: liandefu@ustc.edu.cn)



Question: Who created the show with Wendy Schaal doing the voice of Francine?

Search[Wendy Schaal]

Wendy Schaal (born July 2, 1954)[1] is an American actress known for her work in Joe Dante films such as Innerspace, The 'Burbs, and Small Soldiers. Her other film credits include starring in films such as Where the Boys Are '84, Creature, Going Under, and Munchies. She had many guest roles in television series of the mid-1980s, most notably as Marilyn Kelsy in Airwolf. **Since 2005 she has primarily worked in voice acting, most notably voicing Francine Smith in the animated comedy television series American Dad!**. Schaal was born in Chicago, Illinois, the daughter of Lois Schaal (née Treacy) and actor Richard Schaal.[1] Schaal's father was married to actress Valerie Harper from 1964 to 1978, during which time Harper was her stepmother.[2] From birth until she was five, Schaal lived with her parents in Crete, Illinois, at which time she moved with her mother to Newport Beach, California after her parents divorced.[3] Schaal studied acting with Viola Spolin in Chicago when she was nine years old, later moving to Wisconsin and then California when she was 11.

1. (Wendy Schaal, voiced for, Francine Smith)

2. (Francine Smith, character in, American Dad!)

Figure 1 Examples of suboptimal decisions and irrelevant information in environmental feedbacks. (a) ScienceWorld. Lack of global environmental information causes failure because of being trapped in a loop or a suboptimal solution. (b) HotPotQA. The information retrieved from Wikipedia is massive. However, only the green sentence is helpful, and the other sentences are task-irrelevant.

acquire additional information through exploration. Addressing these challenges requires the agent to perform multistep exploration and exploitation. Exploration involves perceiving the environment and gathering task-relevant information, while exploitation involves making action decisions based on existing knowledge. In existing prompt-based methods, issues of exploration and exploitation are often overlooked, remaining embedded within the reasoning process of the LLM [7] and leading to two major problems.

First, the lack of global awareness of the environment at the outset results in suboptimal decision-making by LLMs. As illustrated in Figure 1(a), the goal is to find an aluminum object and test its conductivity. The agent initially starts outside. The optimal trajectory, shown as the white line, involves the agent first going to the kitchen to take the aluminum fork before going to the workshop. Without global environmental information, the agent is likely to get trapped in a room due to failing to locate an aluminum object (red line) or to take a more time-consuming route (blue line). Second, the knowledge acquired by LLMs through environmental exploration tends to be excessive, including task-irrelevant information. Such irrelevant information disrupts LLM decision-making and incurs additional computational costs. As shown in Figure 1(b), environmental feedback usually contains massive task-irrelevant information, while only one sentence (green line in this example) helps solve the task. This increases token usage and negatively affects the agent's ability to make optimal decisions.

To address the above limitations, we propose a novel prompt-based strategy to enhance LLM agents, termed weak exploration to strong exploitation (WESE). To tackle the first limitation, we propose to decouple exploration and exploitation. Specifically, we construct two distinct LLM agents for these tasks. In the exploration task, the LLM agent interacts with the environment, exploring potentially helpful environmental information for task resolution. In the exploitation task, the information obtained during exploration serves as a global environmental prior, aiding the LLM agent in reasoning and decision-making. To address the second limitation, we compress the environmental information acquired by the exploration agent into a structured knowledge graph. During exploitation, we adopt a one-hop knowledge retrieval approach, selecting one-hop neighbors of task-relevant entities from the graph as priors, thereby reducing interference from irrelevant information. Furthermore, to minimize resource consumption, we observe that a cost-effective, weaker LLM (such as a 7 B model) is sufficient for less challenging exploration tasks. Therefore, we propose a textitweak exploration to strong exploitation strategy—leveraging the knowledge explored by the weaker LLM agent to enhance the performance of the stronger LLM agent.

Our main contributions are summarized as follows.

• To the best of our knowledge, this is the first work to investigate the effect of decoupling exploration and exploitation for LLM agents in open-world tasks. We further propose WESE, which leverages a weaker agent to enhance the stronger agent in a cost-effective manner.

• To better utilize environmental information obtained during exploration, we introduce a strategy to compress it into a knowledge graph, followed by a one-hop retrieval approach to filter out irrelevant information.

• Experimental results over four open-world interactive benchmarks demonstrate the superiority of WESE, achieving a notable balance between effectiveness, efficiency, and cost.

2 Related work

In this section, we introduce the studies for LLM agents and LLM for open-world tasks.

2.1 LLM agents

With the emergence of LLMs, their intelligence has sparked considerable potential in applying LLMs as the brains of agents. Existing LLM agent studies primarily consider three key modules: planning, tool usage, and memory [2]. Planning module [4] aims to empower agent with the task-decomposition ability, encompassing work on task decomposition [13], feedback-driven adjustments [9], and multi-path reasoning [14, 15]. For instance, Plan-and-solve [13] decomposes task into subgoals. LLM+P [16] leverages LLM to convert the task into the PDDL and uses symbolic planner to solve. Tree-of-thought [14] elicits LLM to generate multiple reasoning path and construct a tree to search. Tool usage aims to strengthen the ability to use external tools [17]. For instance, Visual ChatGPT [18] incorporates visual models as tools to augment the LLM's visual capabilities. ToolLlama [19] fine-tunes Llama's ability to leverage various APIs. The memory module focuses on storing feedback information perceived from the environment, assisting the agent with experience, and fostering the growth of the agent. In generative agents [20], memories of simulated roles are stored as texts, utilizing RAG for relevant pieces. REMEMBER [21] proposes a semi-parametric memory, i.e., the Q-value table, to record rewards as the value and action in a given environment and task as the key. MemoryBank [22] leverages the Ebbinghaus forgetting curve, incorporating update and forgetting mechanisms into the memory design.

In this paper, in our proposed WESE, the knowledge graph is essentially a memory, updating information obtained through exploration into the graph.

2.2 LLM for open-world tasks

Open-world tasks represent the simulation of real-world environments. Within these tasks, agents engage in continuous interactions with the environment to gather pertinent information, subsequently making decisions and taking action to accomplish goals. Open-world tasks typically exhibit fewer constraints on the process, placing greater emphasis on the final rewards. Representative examples of open-world tasks include games like "Minecraft" [23, 24], where textual information and visual feedback are involved. Another category comprises text-based simulators based on the TextWorld [10], such as AlfWorld [11], which involves household tasks, ScienceWorld [12], which involves simple scientific experiments, and question-answering tasks [25, 26] where agents need to interact with the web to obtain supporting information, such as Wikipedia. In tackling such tasks, chain-of-thought (CoT) [5] proposes adding few-shot examples in the prompt, guiding the LLM to solve the task step by step. ReAct [7] induces the reasoning capability of LLMs by introducing an extra thought step. Subsequent methods have built upon ReAct, with enhancements such as the Reflexion [9] mechanism, allowing agents to learn from mistakes in subsequent attempts. Additionally, several methods leverage the coding capabilities of LLMs, transforming tasks into programming tasks and guiding LLMs to generate codes as plans, such as VOYAGER [23].

3 Methodologies

3.1 Decoupling exploration and exploitation

Open-world tasks differ from traditional reasoning and decision-making tasks. Traditional reasoning [27, 28] or decision-making [29] tasks typically present all relevant information at once, requiring the agent to deduce and make a plan based on the provided information, such as mathematical calculations or logical reasoning problems. Conversely, in open-world tasks, only the task description is initially specified. In this context, the agent must continually interact with the environment to obtain supporting information, comprising the exploration and exploitation steps.

Let E and T represent the environment and the task, Θ denote the LLM, and P denote the prompt. The action space of the agent is defined as $\mathcal{A} = \mathcal{A}_e \cup \mathcal{A}_t$, where $\mathcal{A}_e$ and $\mathcal{A}_t$ represent the action set of exploration and exploitation, respectively. Exploration and exploitation are defined as the functions $explore(\cdot)$ and $exploit(\cdot)$. The information given by the environment in the i-th step is defined as F_i. Regarding existing methods such as ReAct [7] where exploration and exploitation steps are embedded in reasoning, the action taken at step i is represented as follows:

$$a_i = reason(E, T, s_{i-1}; \Theta, P, K = \cup_{j<i}\{F_j\}) \in \mathcal{A}_e \cup \mathcal{A}_t,$$

where $reasion(\cdot)$ denotes the mixture of exploration and exploitation.

Within this paradigm, the knowledge K utilized is solely the limited information about the environment obtained through partial observations. Particularly, greedy decisions are taken in the initial steps when the agent possesses limited awareness of the environment. For instance, in a task such as "*cleaning some apples with soap*" and the

```
Interact with a household to explore more information for the task. Here are two examples.

You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a
cabinet 3, a cabinet 2, a cabinet 1, a countertop 1, a garbagecan 1, a handtowelholder
2, a handtowelholder 1, a sinkbasin 2, a sinkbasin 1, a toilet 1, a toiletpaperhanger 1,
and a towelholder 1.
Your task is to: put some spraybottle on toilet.
> think: My goal is to explore the room to get more information about the task. In this
task, two entities are involved, they are sparybottle and toilet. The toilet is already
seen, so I need to explore more information about the spraybottle.
...
> think: Now I find a spraybottle (2) and the toilet. My goal is completed.
> Finish

{example 2}

Here is the task.
{task description}
>
```

Figure 2 Prompts for exploration task on ALFWorld. The system instruction and few-shot demonstrations both motivate the LLM to explore more information.

```
Leverage the empirical memory and interact with a household to solve a task. Here are two
examples.

Your empirical memory for the task: [('cup 1', 'on', 'sidetable 1'), ('apple 3', 'on',
'garbagecan 1'), ('lettuce 1', 'on', 'sidetable 1'), ('peppershaker 3', 'on', 'sidetable
1')].
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a
cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 1, a diningtable
3, a diningtable 2, a diningtable 1, a drawer 1, a fridge 1, a garbagecan 1, a microwave
1, a sidetable 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a
stoveburner 1, and a toaster 1.
Your task is to: find some apple and put it in sidetable.
> think: To solve the task, I need to find and take an apple, then put it in sidetable.
First I need to find an apple. According to the memory ('apple 3', 'on', 'garbagecan 1'),
an apple is more likely to appear on garbagecan 1. I can check it firstly and then check
others.
...

{example 2}
Here is the task.
Your empirical memory for the task: {relevant triplets}.
{task description}
>
```

Figure 3 Prompts for exploitation task on ALFWorld. The task is "put" task. The LLM is provided with the empirical memory explored in the exploration stage.

agent's initial location is the hall. The actual locations of the apple and soap are in the drawer of the table in the hall and on the sink in the kitchen, respectively. The lack of environmental knowledge may lead the agent to be misled by the world knowledge of the LLM, going to the kitchen to find the apple. Consequently, substantial efforts traversing every corner of the kitchen are wasted, resulting in suboptimal plans and even failures due to trapping in the loop. Therefore, we investigate the strategy to decouple exploration and exploitation, formalized as follows:

$$a_i = \begin{cases} \mathit{explore}(E, T, s_{i-1}; \Theta, P_e) \in \mathcal{A}_e,\ i < N_e, \\ \mathit{exploit}(E, T, s_{i-1}; \Theta, P_t, K) \in \mathcal{A}_t, i \geqslant N_e, \end{cases}$$

where P_e and P_t represent the prompts of the exploration and exploitation task, respectively. $K = \cup_{j \leqslant N}\{F_j\}$ denotes the knowledge graph obtained in exploration stage. N_e is the maximum number of steps of exploration, which could also be determined by the agent, such as terminating the exploration automatically when it thinks the obtained information is sufficient. We illustrate the prompts for exploration and exploitation on ALFWorld in Figures 2 and 3, respectively.

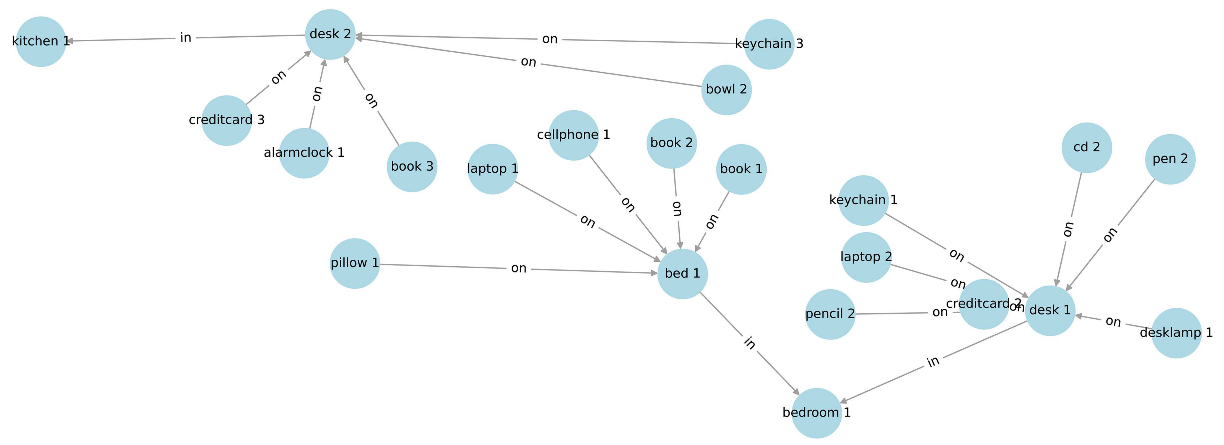


Figure 4 Visualization of the explored knowledge graph in an ALFWorld householding task. Each node represents an object in the house, and the edge denotes the relationship between two objects.

Different from the previous methods, our method places the whole exploration phase before exploitation explicitly, as opposed to the alternation of exploration and exploitation. In this manner, the agent has extensively explored the environment, acquiring global environmental prior knowledge denoted as $K = \cup_{j\leqslant N}\{F_j\}$. Exploitation with global knowledge benefits the effectiveness and efficiency of the solutions, which is empirically validated in our experiments.

However, two subsequent issues exist following the decoupling approach. First, the information obtained from environmental feedback is huge due to the extensive exploration, including a lot of task-irrelevant information. Second, the extensive exploration contributes to increased resource consumption, such as token usage. Therefore, we demand an efficient mechanism for information transfer between exploration and exploitation and a cost-efficient exploration-exploitation strategy. We address the two issues in the subsequent parts of this section.

3.2 Knowledge compression and retrieval

Real-world textual information exhibits inherent sparsity, characterized by long sentences consisting of plenty of non-informative conjunctions and adjectives. Environmental feedback in open-world tasks manifests as such text, where the cumulative extensive exploration yielded long and unstructured textual information, demonstrating severe sparsity. Considering the limited context window of the LLM and the expensive cost of token usage, it is necessary to compress the sparse information. Leveraging a knowledge graph (KG) to store information has proved advantageous in enhancing information density and leveraging domain-specific knowledge in existing studies [30].

Benefiting from the intelligent ability of LLM in knowledge extraction tasks [31], a previous study [32] has demonstrated the superiority of LLMs in triplets extraction, achieving a 5.1% improvement compared with the best finetuned BiLSTM-base method in terms of F1 score on WebNLG dataset. Consequently, we extract the knowledge from the received feedback to form an environmental knowledge graph. Specifically, the LLM extracts knowledge triplets from the environmental feedback after each exploration step, updating them into the knowledge graph. For example, as for the search result given by Wikipedia "*Since 2005 Wendy Schaal has primarily worked in voice acting, most notably voicing Francine Smith in the animated comedy television series American Dad!*", knowledge triplets are extracted as ⟨*Wendy Schaal*, *voice for*, *Francine Smith*⟩ and ⟨*Francine Smith*, *character in*, *American Dad!*⟩. Notably, the environmental knowledge graph we obtained is task-relevant, serving as a memory like the random access memory (RAM). Actually, advanced knowledge extraction techniques and a worldwide knowledge graph could be leveraged and continually in our method, serving as a general memory. We leave it for further work. To provide a better understanding for the extracted knowledge graph, we visualize the graph explored in one householding task ALFWorld [11], presented in Figure 4.

Nevertheless, it is imperative to acknowledge that not all information in the knowledge graph proves useful. The introduction of task-irrelevant information has the potential to lead the hallucination phenomena of LLM, such as the confusion of entity and relation. For example, giving the triplet ⟨*Bob*, *favorite fruit*, *apple*⟩ and the question is "*What's the favorite fruit of Bill?*", the LLM would confuse the relation and answer with *apple*. Benefiting from the graph structure, we adopt a one-hop retrieval method to extract task-related information easily, illustrated in Algorithm 2. Concretely, we initiate the process by extracting involved entities from the task description with LLM. Subsequently, we perform a one-hop retrieval on the graph to obtain the neighbors of these entities. The retrieved

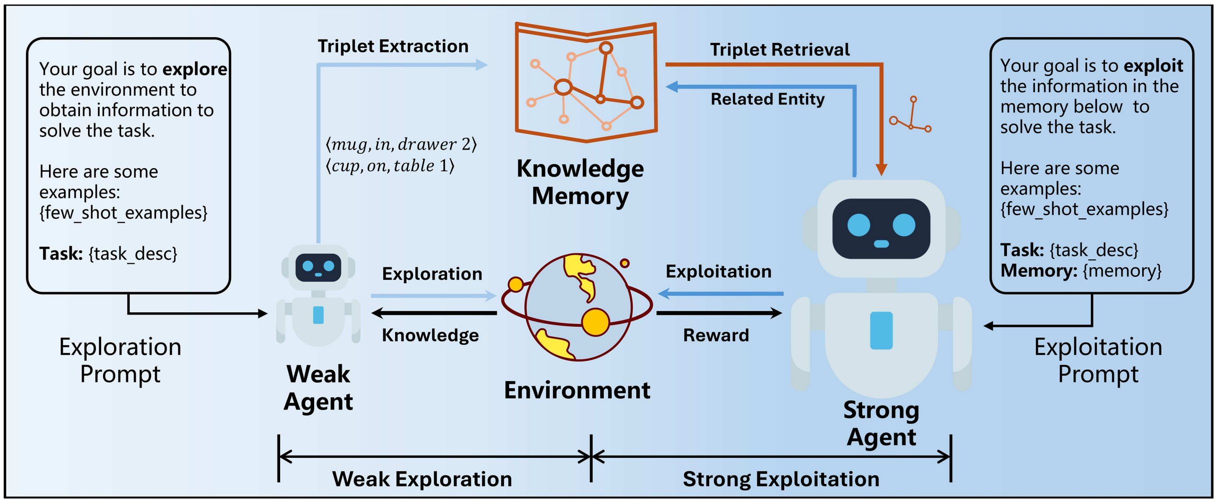


Figure 5 Framework of WESE. In the weak exploration stage (left part), the weaker agent interacts with the environment to obtain more information and construct a knowledge graph. In the strong exploitation stage (right part), the stronger agent retrieves relevant triplets to solve the task. We employ Llama-2-7B as the weak agent and text-davinci-003 as the strong agent in the implementation.

knowledge triplets are then injected into the prompt, serving as task-relevant knowledge during the exploitation phase, thereby assisting the LLM in task-solving.

3.3 Weak exploration to strong exploitation

Acquiring more comprehensive global information about the environment demands a considerable resource cost in the exploration process. However, compared to exploitation, exploration exhibits lower complexity, requiring less reasoning and induction. Concretely, exploration operations exhibit low requirements for the logic and coherence of actions, emphasizing actions pertaining to environmental observation. For example, the exploration actions mainly consist of several simple actions on decision-making benchmarks, such as "*go to* [ROOM]", "*look around*", while exploitation involves a series of coherent operations like (*go to sink/stove, put the bowl in/on the sink/stove, activate the sink/stove, wait, deactivate the sink/stove*).

Therefore, to avoid using a sledgehammer to crack a nut, we propose to use a weaker agent for the less challenging exploration to mitigate resource consumption, namely the weak exploration. From the perspective of the LLM agent, a weaker agent represents substituting the underlying LLM for exploration with a weaker LLM, i.e., an LLM with fewer parameters, thereby reducing costs. In our experiments, we compare performance between strong exploration and weak exploration. Our findings reveal that a weaker exploration has a negligible impact on the final success rate, yet it significantly lowers costs.

The framework of WESE is illustrated in Figure 5. There are three key components in the framework: a weak LLM agent, a strong LLM agent, and a KG-based memory. The whole process consists of the weak exploration (left) and the strong exploitation (right). Meanwhile, we offer an algorithmic pseudo-code in Algorithm 3. First, a weak LLM agent is employed to explore the interactive environment to obtain information in lines 1–11, where the maximum step of exploration is set as N_e. Additionally, an extra "COMPLETE" action is added to the action spaces via a prompt, allowing the weak agent to stop exploring adaptively when it determines there is sufficient knowledge about the task. Then those knowledge triplets are organized as a knowledge graph G_K in line 12, as illustrated in Algorithm 1. Consistent with the setting in Reflexion [9], the state is then set as the initial state s_0 at the start of exploitation in line 13. Further, the involved entities are extracted from the task with an LLM and the relevant triplets are retrieved from the graph in line 14. Retrieved knowledge is leveraged for exploitation in line 16, serving as prior knowledge.

4 Experiments

We employ two categories of interactive open-world tasks as benchmarks: decision-making and question-answering, where each task requires multi-step interactions with the environment. We evaluate our methods from three perspectives: effectiveness, efficiency, and cost, representing whether the agent can complete the tasks, how many

Algorithm 1 Graph construction algorithm.

```
Input: Knowledge triplets set $K$;
Output: Knowledge graph $G$;
 1: Entity set $E \leftarrow \{\}$;
 2: Relation set $R \leftarrow \{\}$;
 3: Adjacency matrix $M$;
 4: for $x \in K$ do
 5:     $h, r, t \leftarrow x$;
 6:     $E \leftarrow E \cup \{h, t\}$;
 7:     $R \leftarrow R \cup \{r\}$;
 8:     $M[h][t] \leftarrow r$;
 9: end for
10: $G.E \leftarrow E$;
11: $G.R \leftarrow R$;
12: $G.M \leftarrow M$;
```

Algorithm 2 Triplet retrieval algorithm.

```
Input: Knowledge graph $G$, Task $T$, LLM $\Theta$;
Output: Triplets set $K$;
 1: Task-related entity set $E \leftarrow extract(G.E, T; \Theta)$;
 2: $K \leftarrow \{\}$;
 3: for $e_i \in E$ do
 4:     $r \leftarrow G.M[e_i][e_j]$;
 5:     if $r \neq$ empty then
 6:         $K \leftarrow K \cup \{(e_i, r, e_j)\}$;
 7:     end if
 8: end for
```

Algorithm 3 WESE algorithm.

```
Input: KEnvironment $E$, task $T$, initial state $s_0$, weak LLM $\Theta_w$, strong LLM $\Theta_s$, exploration prompt $P_e$, exploitation prompt $P_t$, limit of
       steps $N_e, N_t$;
Output: Plan $p$;
                                                        ▷ Stage1: Exploration with weak LLM agent.
 1: $K \leftarrow \{\}$; $i \leftarrow 0$; $s_i^e \leftarrow s_0$;
 2: for $i < N_e$ do
 3:     $a_i^e \leftarrow explore(E, T, s_i^e; \Theta_w, P_e)$;
 4:     if $a_i^e \neq$ COMPLETE then
 5:         break;                  ▷ Adaptive stop of exploration according to the weak agent.
 6:     end if
 7:     $s_i^e, F_i \leftarrow step(E, s_i^e, a_i^e)$;
 8:     $K' \leftarrow extract(F_i; \Theta_w)$;
 9:     $K \leftarrow K \cup K'$;
10:     $i \leftarrow i + 1$;
11: end for
12: $G_K \leftarrow construct\_graph(K)$;                                  ▷ Refer to Algorithm 1.
                                                        ▷ Stage2: Exploitation with strong LLM agent.
13: $i \leftarrow 0$; $s_i^t \leftarrow s_0$; $p \leftarrow []$;
14: $\tilde{K} \leftarrow retrieve\_triplets(G_K, T; \Theta_w)$;                ▷ Refer to Algorithm 2.
15: for $i < N_t$ and $F_i \neq$ Completed do
16:     $a_i^t \leftarrow exploit(E, T, s_i^t; \Theta_s, P_t, \tilde{K})$;
17:     $s_i^t, F_i \leftarrow step(E, s_i^t, a_i^t)$;
18:     $i \leftarrow i + 1$;
19:     $p \leftarrow p + [a_i^t]$;
20: end for
```

steps the agent would take to finish the task, and the expenses for the agent to complete the task, respectively. The implementation of WESE will be released upon acceptance.

4.1 Decision making tasks

We begin with the open-world decision-making tasks, where environments are based on a text-based simulator. The tasks are about the household, where the agent needs to explore various rooms and take operations on several objects.

4.1.1 *ALFWorld*

ALFWorld [11] is a synthetic text-based simulated interactive environment. It comprises six types of tasks where agents need to interact with the environment to generate a series of actions to solve household tasks. For example, in the task "clean some knife and put it in countertop", the ideal solution involves actions such as (go to countertop 2, take knife 1, go to sinkbasin 2, clean knife 1, put knife 1 on countertop 2). These tasks vary in difficulty, with challenging tasks encompassing over 50 locations and requiring more than 50-step actions, posing challenges for both the exploration and exploitation processes.

To validate the effectiveness of WESE, we adopt Act [7], ReAct [7], and Reflexion [9] as baselines. Besides, AgentTuning [33] and AgentEvol [34] are implemented as the representation of fine-tuning method[1)], which trains the LLM with the agent data. We use the weights released by [33] in huggingface[2)] Act leverages the idea of CoT, providing LLMs with few-shot interactive examples. ReAct, building upon Act, introduces an extra "THOUGHT" step where LLMs can choose to explicitly output their thought about the current state or generate an action. Reflexion introduces a correction mechanism based on ReAct, which reflects on previous failures and uses those reflections as prompts in the next trial. Here we set the number of reflection steps for each task as two, which means there are at most two trials for each task. In WESE, we initially use a weak LLM for exploration to acquire task-relevant knowledge. We then leverage the obtained knowledge to solve problems with two base methods. We employ Llama-2-7B [35] as the weak LLM and text-davinci-003 (with probably more than 175 billion parameters) developed by OpenAI[3)] as the strong LLM. The limits of steps N_e, N_t are both set to 50. Our evaluation focuses on success rates, average steps to complete tasks, and the cost of OpenAI API tokens as three key metrics. Additionally, we introduce a variant of WESE—strong exploration to strong exploitation (SESE), where the weak LLM in the exploration process is replaced with the strong LLM, to verify the effectiveness of the decoupling strategy and examine the impact of LLM strength on exploration quality.

4.1.2 *ScienceWorld*

Similar to ALFWorld, ScienceWorld [12] is an interactive household environment as well. However, the tasks in ScienceWorld are more challenging, involving scientific experiments such as boiling and creating a new color by mixing primary colors. The environment is more complex, comprising ten distinct rooms, each with different furnishings, and not each pair of rooms is connected.

We conduct experiments on eight types of tasks within ScienceWorld, choosing about 30 instances for each task due to a limited budget. Unlike ALFWorld where the agent can get a reward of 1 only when the task is completed, the agent in ScienceWorld receives partial rewards upon completing crucial steps, with the total reward reaching 100. Given the challenging nature of the tasks, achieving a full reward of 100 is rare. Therefore, we utilize the number of steps taken by the agent until it first obtains a positive reward as the metric for efficiency. Other settings are consistent with ALFWorld.

4.1.3 *Results*

The results on ALFWorld and ScienceWorld are shown in Tables 1 and 2, respectively. We conclude several findings based on the results. Consistent with results reported in ReAct [7], ReAct outperforms Act on two benchmarks, showing the superiority of the "THOUGHT" step. However, this additional step leads to a longer action sequence, resulting in an average relative 32.38% increase in average steps. Benefiting from more trials and reflections, Reflexion shows superiority compared with ReAct, resulting in a higher success rate while much more cost.

By fine-tuning the 7B LLM with the agent data collected from various benchmarks, such as WebShop and ALFWorld, AgentTuning shows competitive performance with the Act method in terms of success rate in ALFWorld. However, for the held-out benchmark SciWorld that the smaller LLM is not trained on, the larger LLM shows significant advantages over AgentTuning, which signifies the weaker generalization in solving those agent tasks for those smaller LLMs. Regarding AgentEvol, which has been fine-tuned on a variety of agent-based tasks, including ALFWorld and SciWorld, this model demonstrates markedly superior performance in these two domains. The observed enhancements are primarily attributable to domain-specific training, as evidenced by the notable decline in performance when evaluated on subsequent question-answering tasks. This suggests that while in-domain training significantly boosts task-specific capabilities, it may not translate into comparable improvements across different task domains.

Decoupling of exploration and exploitation demonstrates advantages in effectiveness and efficiency, resulting in SESE outperforming baselines significantly with average relative 21.80% and 25.34% improvements in terms of success rate (average reward) and average steps. However, the cost of SESE increases a lot due to the introduction

1) Since this kind of method requires training of LLMs, the cost calculated by tokens is not applicable. We mark it as NA in subsequent tables.

2) https://huggingface.co/THUDM/agentlm-7b.

3) https://platform.openai.com/.

Table 1 Results on ALFWorld (134 tasks). SR and AS are abbreviations for success rate and average steps of successful tasks, respectively. SESE represents the variant of WESE—strong exploration to strong exploitation. The *Imp* represents the relative improvements compared to base methods, i.e., Act and ReAct. The best and the second best results for the same base method are in bold and underlined, respectively. The up/down arrow indicates that a higher/lower value is better.

Method	Effectiveness		Efficiency		Cost			
	SR↑	*Imp* (%)	AS↓	*Imp* (%)	Prompt↓	Completion↓	Expense ($)↓	*Imp* (%)
AgentTuning	0.42	NA	15.55	NA	NA	NA	NA	NA
AgentEvol	0.44	NA	11.46	NA	NA	NA	NA	NA
Act	0.43	0.00	10.83	0.00	4908548	21243	98.60	0.00
Act-WESE	0.63	+46.51	7.54	+30.38	3746290	19562	**75.32**	+23.61
Act-SESE	**0.67**	+55.81	**6.73**	+37.86	7259508	75153	146.69	−48.77
ReAct	0.57	0.00	16.64	0.00	7565676	43250	152.18	0.00
ReAct-WESE	0.72	+26.32	13.69	+17.73	5032374	41004	**101.47**	+33.32
ReAct-SESE	**0.75**	+31.58	**12.41**	+25.42	8996182	97286	181.87	−19.51
Reflexion	0.71	0.00	14.02	0.00	10934543	73902	220.17	0.00
Reflexion-WESE	0.75	+5.63	12.65	+9.77	10162891	71059	**204.68**	+7.04
Reflexion-SESE	**0.78**	+9.86	**12.02**	+14.27	14210517	126786	286.75	−30.24

Table 2 Results on ScienceWorld (296 tasks). TR, AR, and AS are abbreviations for total reward, average reward, and average steps to get positive reward, respectively. SESE represents the variant of WESE—strong exploration to strong exploitation. The *Imp* represents the relative improvements compared to base methods, i.e., Act and ReAct. The best and the second best results for the same base method are in bold and underlined, respectively. The up/down arrow indicates that a higher/lower value is better.

Method	Effectiveness			Efficiency		Cost			
	TR↑	AR↑	*Imp* (%)	AS↓	*Imp* (%)	Prompt↓	Completion↓	Expense ($)↓	*Imp* (%)
AgentTuning	3395	11.47	NA	23.42	NA	NA	NA	NA	NA
AgentEvol	7363	24.88	NA	8.22	NA	NA	NA	NA	NA
Act	4908	16.58	0.00	18.00	0.00	13554960	55817	272.22	0.00
Act-WESE	5198	17.56	+5.91	15.68	+12.91	13491043	65952	**271.14**	+0.40
Act-SESE	**5249**	**17.73**	+6.94	**15.39**	+14.49	36424190	165568	731.80	−168.83
ReAct	4454	15.05	0.00	20.00	0.00	17716698	84724	356.03	0.00
ReAct-WESE	**5317**	**17.96**	+19.34	19.65	+1.77	16310632	80851	**327.83**	+7.92
ReAct-WESE	5053	17.07	+13.42	**19.02**	+4.92	40293571	196338	809.80	−127.45
Reflexion	5021	16.96	0.00	19.32	0.00	29479402	149204	592.57	0.00
Reflexion-WESE	5693	**19.23**	+13.38	**19.21**	+0.57	25203120	119820	**506.46**	+14.53
Reflexion-SESE	5682	19.20	+13.16	19.25	+0.36	46293102	219102	930.24	−56.98

of extensive strong exploration, showing an average relative 75.30% increase of cost over baselines. Notably, even Reflexion may conduct two trials on one task, the reflections on the failure are not global environmental information, thereby our WESE and SESE both make significant improvements.

WESE shows a better balance between effectiveness, efficiency, and cost, which saves 48.25% of costs with only relative 1.57% and 5.43% degradations in effectiveness and efficiency compared with SESE. In WESE, the weak LLM agent undertakes the exploration process, resulting in cost savings for extensive exploration. Besides, benefiting from the related triplets extracted from the explored KG, the strong LLM agent only needs to focus on exploitation, further decreasing the number of steps, evidenced by the decreased completion tokens and average steps.

We further investigate the improvements of WESE on various types of tasks on ALFWorld, shown in Figure 6. Both WESE and SESE show improvements over almost all types of tasks, further indicating the effectiveness of the decoupling strategy. In addition, the improvements in "*clean*" and "*heat*" tasks are greater than other tasks. The reason lies in that the two tasks involved more complicated exploitation compared with "*put*", where the agents need to find the object first and then clean or heat it instead of just moving it to another place. The result demonstrates that extensive exploration benefits more for complex tasks.

4.2 Question answering tasks

We also validate our WESE on two open-world interactive question-answering benchmarks, i.e., HotPotQA and FEVER. Different from traditional question-answering tasks where supporting sentences are given, those tasks provide the question only and require the agent to search for information on the web step by step to give the final

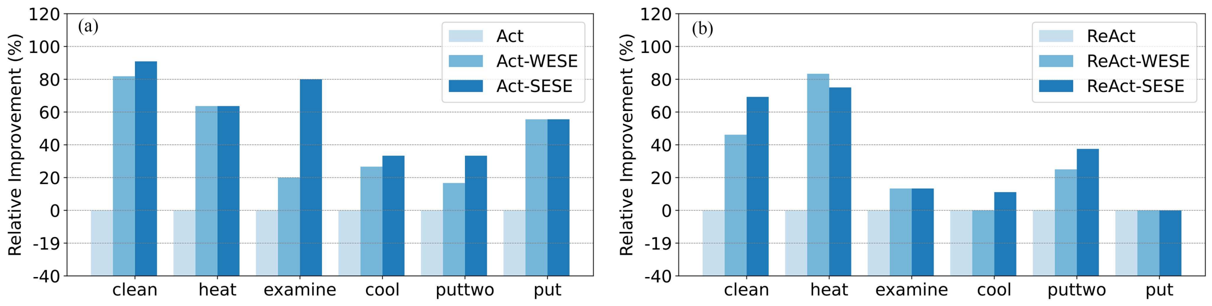


Figure 6 Relative improvements in success rate over various types of tasks on ALFWorld. The left tasks are more complicated than the right. (a) Relative improvements for Act-based methods; (b) relative improvements for ReAct-based methods.

answer.

4.2.1 *HotPotQA*

HotPotQA [25] is a question-answering dataset where each question is paired with supporting sentences from Wikipedia articles. In traditional QA tasks, the supporting sentences are given and the remained task is to reason. Referred in ReAct, we use the Wikipedia API with three types of actions to support interactive information retrieval: (1) SEARCH[ENTITY], which searches the Wikipedia with the ENTITY and returns the corresponding page if it exists, or suggests top-5 similar entities; (2) LOOKUP[KEYWORD], which looks up keyword in the page and returns the next sentence containing the KEYWORD, simulating the Ctrl+F function in a web browser; (3) FINISH[ANSWER], which answers the question with ANSWER. Once the ANSWER matches the ground truth, the environment would return reward 1. We sample 500 tasks from the development set.

We employ the CoT [5], Act, and ReAct as baselines and empower Act, ReAct, and Reflexion with WESE and SESE. Note that CoT is a one-step method that does not support interactive tasks. We inject the supporting sentences into the prompts and instruct the LLM to reason for the final answer without searching on the web. Also, WESE is not designed for such a purely reasoning method but for methods involving interactions with the environment. In addition, two fine-tuning-based methods are employed as baselines: AgentTuning [33] and AgentEvol [34]. For Act and ReAct, we keep the settings consistent with the original paper. For Reflexion, we set the maximum number of trials as 2. As there are probably lots of related triplets to the task-involved entities, we set the limit of retrieved triplets as 10 and the limits of steps N_e, N_t as 8. The evaluation for effectiveness, efficiency and cost is consistent with the ALFWorld.

4.2.2 *FEVER*

FEVER [26] is a fact verification dataset, consisting of instances where each instance comprises a claim and a justification (TRUE or FALSE or NOT CLEAR). We employ the Wikipedia API to construct an interactive environment consistent with that in HotPotQA. Other settings are kept consistent with HotPotQA, such as the number of retrieved triplets and the maximum steps.

4.2.3 *Results*

The results on HotPotQA and FEVER are shown in Tables 3 and 4, respectively. We can draw several findings based on the results. Similar to decision-making tasks, ReAct outperforms Act significantly due to the additional "THOUGHT" step. And Reflexion improves ReAct with one more trial. For the fine-tuning method AgentTuning, since the model is not directly trained on samples in HotpotQA and FEVER, the performance is far from the prompt-based methods. Regarding AgentEvol without tuning on any question-answering tasks, it shows almost zero success rate in both HotpotQA and FEVER, which could be attributed into two reasons: first, the supervised fine-tuning on other tasks harms the model's generalization ability; second, the limited context length of the model (2048 tokens) is not enough to solve the QA tasks, in which long observation is derived from the Wikipedia search engine.

Furthermore, methods equipped with WESE or SESE outperform baselines in both success rate and the number of taken actions, resulting in average relative improvements of 11.77% and 23.86%, respectively. Especially, SESE methods surpass WESE slightly with average relative 4.35% and 3.72% improvements in terms of success rate and average steps, while increasing more than twice the expenses. This further demonstrates that the weak agent powered by Llama-2-7B is almost sufficient for the exploration task.

Table 3 Results on HotPotQA (500 tasks). SR and AS are abbreviations for success rate and average steps of successful tasks, respectively. SESE represents the variant of WESE—strong exploration to strong exploitation. The *Imp* represents the relative improvements compared to base methods, i.e., Act, ReAct, and Reflexion. The best and the second best results for the same base method are in bold and underlined, respectively. The up/down arrow indicates that a higher/lower value is better.

Method	Effectiveness		Efficiency		Cost			
	SR↑	*Imp* (%)	AS↓	*Imp* (%)	Prompt↓	Completion↓	Expense ($)↓	*Imp* (%)
AgentTuning	0.18	NA	3.67	NA	NA	NA	NA	NA
AgentEvol	0.01	NA	4.14	NA	NA	NA	NA	NA
CoT	0.32	N/A	1.00	N/A	261347	25382	5.73	N/A
Act	0.30	0.00	3.53	0.00	2390041	14236	<u>48.09</u>	0.00
Act-WESE	<u>0.35</u>	+19.26	<u>2.69</u>	+23.80	2307421	13973	**46.42**	+3.45
Act-SESE	**0.36**	+21.96	**2.58**	+26.91	7522826	271551	155.89	−224.18
ReAct	0.34	0.00	3.17	0.00	3234876	65306	<u>66.00</u>	0.00
ReAct-WESE	<u>0.39</u>	+15.20	<u>2.29</u>	+27.76	2574401	67908	**52.85**	+19.93
ReAct-SESE	**0.42**	+21.64	**2.11**	+33.44	7338590	323401	153.24	−132.17
Reflexion	0.39	0.00	2.35	0.00	5530021	94421	<u>112.49</u>	0.00
Reflexion-WESE	<u>0.40</u>	+3.06	<u>2.18</u>	+7.23	4638921	90327	**94.58**	+15.92
Reflexion-SESE	**0.42**	+7.14	**2.05**	+12.77	9386284	338921	194.50	−72.91

Table 4 Results on FEVER (500 tasks). SR and AS are abbreviations for success rate and average steps of successful tasks, respectively. SESE represents the variant of WESE—strong exploration to strong exploitation. The *Imp* represents the relative improvements compared to base methods, i.e., Act, ReAct, and Reflexion. The best and the second best results for the same base method are in bold and underlined, respectively. The up/down arrow indicates that a higher/lower value is better.

Method	Effectiveness		Efficiency		Cost			
	SR↑	*Imp* (%)	AS↓	*Imp* (%)	Prompt↓	Completion↓	Expense ($)↓	*Imp* (%)
AgentTuning	0.31	NA	3.24	NA	NA	NA	NA	NA
AgentEvol	0.00	NA	9.00	NA	NA	NA	NA	NA
CoT	0.61	N/A	1.00	N/A	100387	11942	2.25	N/A
Act	0.56	0.00	2.16	0.00	723646	6980	<u>14.61</u>	0.00
Act-WESE	<u>0.62</u>	+10.71	<u>1.58</u>	+26.66	723867	5937	**14.60**	+0.11
Act-SESE	**0.64**	+14.29	**1.57**	+27.34	2822189	122543	60.89	−316.73
ReAct	0.63	0.00	2.18	0.00	1074080	36040	<u>22.20</u>	0.00
ReAct-WESE	<u>0.68</u>	+7.26	<u>1.62</u>	+25.96	918905	29895	**18.98**	+14.53
ReAct-SESE	**0.70**	+10.09	**1.59**	+27.18	3104924	162363	65.35	−194.32
Reflexion	0.66	0.00	2.05	0.00	1795093	67774	<u>37.26</u>	0.00
Reflexion-WESE	<u>0.68</u>	+3.03	<u>1.58</u>	+22.93	1752904	68109	**36.42**	+2.25
Reflexion-SESE	**0.71**	+7.58	**1.55**	+24.39	3910432	21892	78.65	−111.09

Different from decision-making tasks, question-answering tasks require fewer steps due to more information being returned with one search action. However, our WESE and SESE are still capable of reducing the number of steps, further showing the advantage of the explored knowledge. As for the cost, the tokens increased in SESE are far more than those in decision-making tasks, which can be attributed to the long-textual feedback from Wikipedia.

4.3 Web browsing tasks

To further verify the effectiveness of our WESE, we conduct experiments on the web-browsing benchmark WebShop. The agent is given an instruction to buy a proper product in an interactive web environment, where the valid actions consist of search and click. Once the agent buys the product that is relevant to the requirements given in the instruction, a positive reward in $[0, 1]$ would be returned according to the relevance. Consistent with the householding tasks, we adopt Act, ReAct, and Reflexion as the baseline methods, and enhance those methods with the proposed WESE and SESE strategies, respectively. Besides, we compare the fine-tuning method AgentTuning as another baseline, where we adopt the weights of the 7B model released by [33]. In our experiments, we set the maximum step for each method as 15. The results are reported in Table 5.

Different from the conclusion in previous sections, Act strategy outperforms other baselines in terms of average rewards and average steps. We notice that the LLM with ReAct prompt would be "overthinking", thereby always trying to change the search keywords instead of checking products when some conditions are not met. For example,

Table 5 Results on WebShop (100 tasks). AR and AS are abbreviations for the average rewards and average steps of successful tasks, respectively. SESE represents the variant of WESE—strong exploration to strong exploitation. The *Imp* represents the relative improvements compared to base methods, i.e., Act, ReAct, and Reflexion. The best and the second best results for the same base method are in bold and underlined, respectively. The up/down arrow indicates that a higher/lower value is better.

Method	Effectiveness		Efficiency		Cost			
	AR↑	*Imp* (%)	AS↓	*Imp* (%)	Prompt↓	Completion↓	Expense ($)↓	*Imp* (%)
AgentTuning	0.42	NA	6.45	NA	NA	NA	NA	NA
AgentEvol	0.64	NA	5.60	NA	NA	NA	NA	NA
Act	0.65	0.00	5.00	0.00	528492	3642	<u>10.64</u>	0.00
Act-WESE	<u>0.67</u>	+3.08	<u>4.87</u>	+2.60	521640	3495	**10.50**	+1.32
Act-SESE	**0.68**	+4.62	**4.83**	+3.40	1203492	11903	24.31	−128.48
ReAct	0.40	0.00	7.72	0.0	1485163	39739	<u>30.50</u>	0.00
ReAct-WESE	<u>0.53</u>	+32.50	<u>6.34</u>	+17.88	1267289	33217	**26.01**	+14.72
ReAct-SESE	**0.56**	+40.00	**6.27**	+18.78	2139496	41374	43.62	−43.02
Reflexion	0.46	0.00	7.48	0.00	2393291	68213	<u>49.23</u>	0.00
Reflexion-WESE	<u>0.54</u>	+17.39	<u>6.14</u>	+17.91	2226902	65418	**45.85**	+6.87
Reflexion-SESE	**0.55**	+19.57	**6.11**	+18.32	3092248	73628	63.32	−28.62

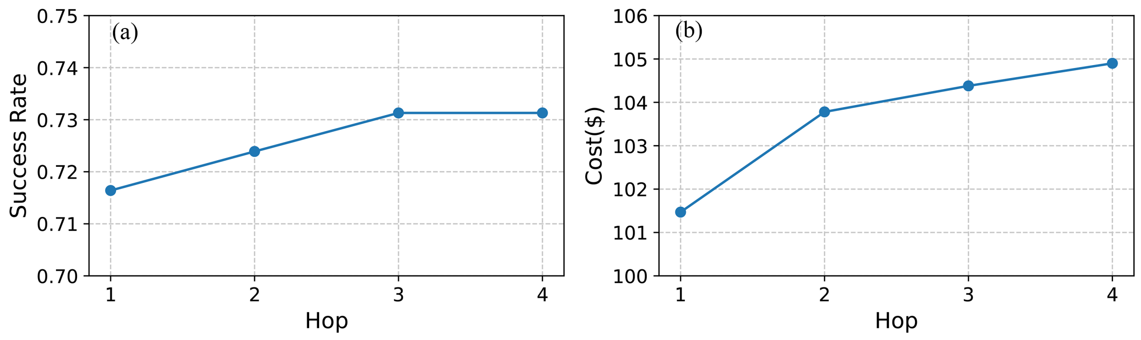


Figure 7 Study on multi-hop retrieval strategy on ALFWorld. The baseline method is ReAct. (a) Success rate; (b) cost.

for the instruction “Find a blue bag under 120 dollars”, the products in the search results may not blue but some of them have the blue options in the detailed page, while the agent refuses to click the product to check optional options but tries to search with other keywords with tiny modifications that have almost no effect on the search results. On the other hand, AgentTuning achieves competitive performance due to its training on the WebShop dataset. Notably, methods equipped with WESE and SESE all show significant improvements in terms of rewards and average steps, demonstrating the robust effectiveness of the decoupling strategy. Similarly, weak exploration can obtain competitive performance compared with strong exploration while reducing the cost obviously.

4.4 Study on multi-hop retrieval

Since the quality of the retrieval strategy can influence the final performance, we conduct experiments with the multi-hop retrieval strategy in the ALFWorld task. For each task, we first extract involved objects and then retrieve all the k-hop neighbors from the graph explored by the weak agent, forming the memory for the exploitation. We set k as 1, 2, 3, 4, respectively. The results are shown in Figure 7.

The results suggest that the accuracy would increase slightly as k increases, while the cost would increase as the triplets given in the prompt increase. This signifies that the task does not require deep knowledge hidden in multi-hop retrieval. We found that most of the tasks in those benchmarks only demand some direct and simple knowledge. For example, for task *put the hot apple on the table*, what the agent needs are the positions of the apple, microwave, and table, while the high-order knowledge is helpless. Therefore, multi-hop retrieval is luxurious for those benchmarks even though it could obtain more high-order knowledge. We believe that high-order knowledge would be of great importance for complex tasks.

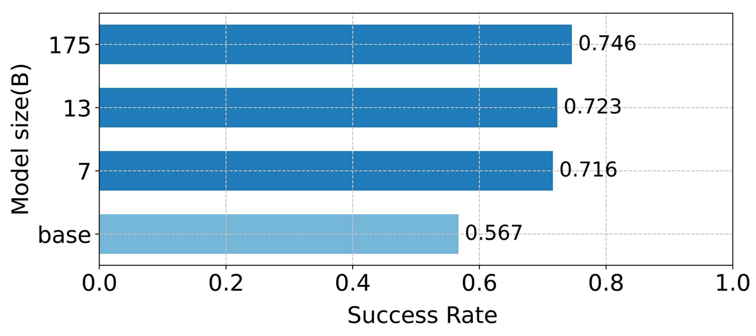


Figure 8 Study on stronger exploration agents on ALFWorld. The baseline method is ReAct.

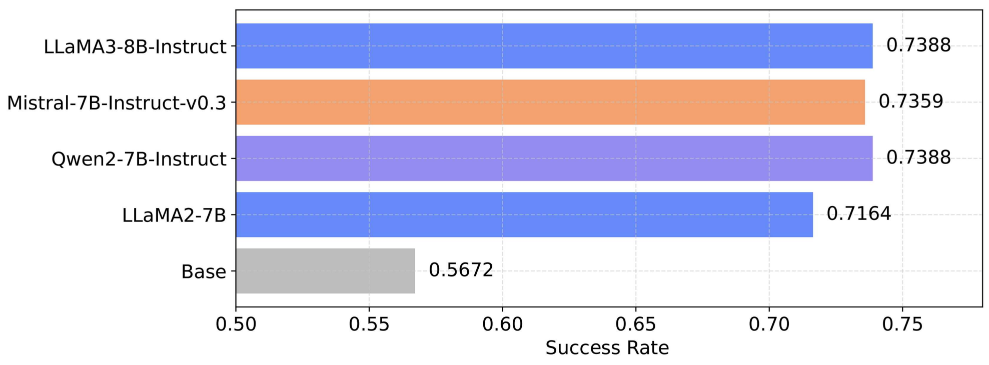


Figure 9 Study on various weak exploration agents on ALFWorld. The baseline method is ReAct.

4.5 Study on various exploration agents

4.5.1 *Stronger exploration agents*

To further validate the effectiveness of the weak exploration, we leverage Llama-2-13B as the base of the exploration agent additionally. We conduct experiments on ALFWorld and report the success rates of three exploration agents with different sizes: 7 B, 13 B, and 175 B[4]. The results are illustrated in Figure 8.

As the model size of the exploration agent increases from 7 B to about 175 B, the improvement of the success rate is limited. Concretely, leveraging a 7 B model as the weak exploration agent could obtain a 26.32% relative improvement compared with the base method. While elevating the parameter count of the exploration agent from 7 billion to 13 billion, the observed relative enhancement amounts merely to 0.97%. Furthermore, upon further augmentation to 175 billion parameters, the relative improvement remains constrained at 4.02%. The results further verify that a smaller LLM is capable of effectively undertaking the exploration task.

4.5.2 *Various weak exploration agents*

To further study the effects of various weak exploration agents, we leverage four different 7 B to 8 B-scale LLMs as the engine of the exploration step, including LLaMA2-7B, Qwen2-7B-Instruct, Mistral-7B-Instruct-v0.3, and LLaMA3-8B-Instruct. The experiments are conducted on the ALFWorld Benchmark and the results are illustrated in Figure 9.

We notice that each weak agent shows significant improvements over the baseline method, further providing a piece of evidence that the weak agent can afford the exploration tasks. And LLaMA3-8B-Instruct [36], Mistral-7B-Instruct-v0.3[5], and Qwen2-7B-Instruct [37] outperform LLaMA2-7B slightly due to more high-quality training and stronger capability in general tasks.

4.6 Cost of weak exploration

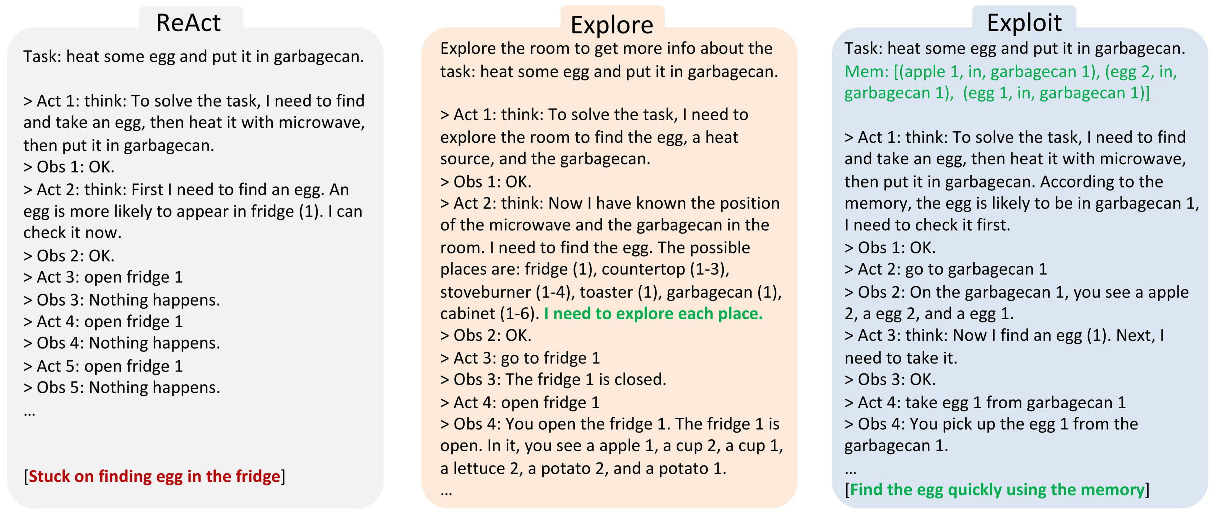
To demonstrate the cost-saving efficacy of our weak exploration strategy, we conducted a comparative analysis between the weak-exploration-specific cost and the strong-exploitation cost, focusing on token consumption and

4) The size of GPT-3.5-turbo is not released; here we assume that GPT-3.5-turbo has the same size as GPT-3.

5) https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3.

Table 6 Cost of weak exploration. The base method is ReAct.

Task	Exploration		Exploitation		Exploration/exploitation
	Prompt	Completion	Prompt	Completion	
ALFWorld	4052238	55840	5032374	41004	0.042
ScienceWorld	18038792	112057	16310632	80851	0.057
HotPotQA	4489137	274271	2574401	67908	0.093
FEVER	2414005	169023	918905	29895	0.141
WebShop	881820	9193	1267289	33217	0.036



ReAct

Task: heat some egg and put it in garbagecan.

> Act 1: think: To solve the task, I need to find and take an egg, then heat it with microwave, then put it in garbagecan.
> Obs 1: OK.
> Act 2: think: First I need to find an egg. An egg is more likely to appear in fridge (1). I can check it now.
> Obs 2: OK.
> Act 3: open fridge 1
> Obs 3: Nothing happens.
> Act 4: open fridge 1
> Obs 4: Nothing happens.
> Act 5: open fridge 1
> Obs 5: Nothing happens.
…

[**Stuck on finding egg in the fridge**]

Explore

Explore the room to get more info about the task: heat some egg and put it in garbagecan.

> Act 1: think: To solve the task, I need to explore the room to find the egg, a heat source, and the garbagecan.
> Obs 1: OK.
> Act 2: think: Now I have known the position of the microwave and the garbagecan in the room. I need to find the egg. The possible places are: fridge (1), countertop (1-3), stoveburner (1-4), toaster (1), garbagecan (1), cabinet (1-6). **I need to explore each place.**
> Obs 2: OK.
> Act 3: go to fridge 1
> Obs 3: The fridge 1 is closed.
> Act 4: open fridge 1
> Obs 4: You open the fridge 1. The fridge 1 is open. In it, you see a apple 1, a cup 2, a cup 1, a lettuce 2, a potato 2, and a potato 1.
…

Exploit

Task: heat some egg and put it in garbagecan.
Mem: [(apple 1, in, garbagecan 1), (egg 2, in, garbagecan 1), (egg 1, in, garbagecan 1)]

> Act 1: think: To solve the task, I need to find and take an egg, then heat it with microwave, then put it in garbagecan. According to the memory, the egg is likely to be in garbagecan 1, I need to check it first.
> Obs 1: OK.
> Act 2: go to garbagecan 1
> Obs 2: On the garbagecan 1, you see a apple 2, a egg 2, and a egg 1.
> Act 3: think: Now I find an egg (1). Next, I need to take it.
> Obs 3: OK.
> Act 4: take egg 1 from garbagecan 1
> Obs 4: You pick up the egg 1 from the garbagecan 1.
…
[**Find the egg quickly using the memory**]

Figure 10 Case study on ALFWorld. In the left part, the agent is stuck finding the egg because it thinks it will be in the fridge. The exploration steps can obtain environmental knowledge, which is leveraged in the exploitation phase, thereby motivating the agent to find the egg quickly.

computational FLOPs (floating-point operations). We instantiated the weak and strong agents with 7 B and 135 B parameter GPT-architecture models, respectively. Existing analysis [38] shows that the FLOPs required for model forward exhibit an approximate linear relationship with the number of model parameters per token. Consequently, the per-token computational cost of the strong agent is estimated to be $135/7 \approx 17.86$ times that of the weak agent. The results of the cost comparison between weak exploration and strong exploitation are detailed in Table 6.

Notably, even when the token consumption during the exploration phase is comparable to that of the exploitation phase, the total cost of weak exploration accounts for merely 5% of the strong exploitation cost. This substantial reduction in computational overhead can be primarily attributed to the inherent efficiency of the weak agent without severely sacrificing the success rate.

4.7 Case study

To provide a more intuitive understanding of our method, we present a simple case for demonstration, as shown in Figure 10.

In such a simple task "heat some egg and put it in garbagecan", the agent with the ReAct prompt fails in finding the egg because it thinks the egg is more likely in the fridge according to its world knowledge, thereby sticking to open the fridge. Instead, our exploration agent could find the egg since its goal is to explore the room, which means it would check each possible place and record the knowledge, serving as the memory of the exploitation agent.

5 Conclusion

In this paper, we introduce WESE, a cost-effective method for enhancing LLM agents in open-world interactive tasks. We decouple exploration and exploitation to acquire more global information about the environment and employ two agents for these distinct processes. To enable effective communication between the two processes, we introduce a knowledge graph-based memory that compresses and structures information obtained during exploration. Task-relevant information is extracted from this graph by a one-hop retrieval method. We further propose leveraging a weaker agent for the exploration process, achieving cost efficiency with negligible performance degradation. Experimental results demonstrate the superiority of WESE in terms of effectiveness, efficiency, and cost.

6 Limitations

Our work has certain limitations and areas for improvement in the following two aspects. First, although the knowledge graph constructed during the exploration phase can be shared across tasks, current benchmarks do not provide shared environment settings, making us unable to experimentally validate this feature. Nevertheless, this does not affect the effectiveness of WESE under the current setting. Second, similar to the Reflexion [9] method, our approach assumes that the environment can be pre-explored. For environments that cannot be explored in advance, the applicability of our method is limited. In future work, we will investigate efficient agent strategies for such environments.

References

1 Zhao W X, Zhou K, Li J, et al. A survey of large language models. 2023. ArXiv:2303.18223

2 Wang L, Ma C, Feng X, et al. A survey on large language model based autonomous agents. 2023. ArXiv:2308.11432

3 Xi Z, Chen W, Guo X, et al. The rise and potential of large language model based agents: a survey. 2023. ArXiv:2309.07864

4 Huang X, Liu W, Chen X, et al. Understanding the planning of LLM agents: a survey. 2024. ArXiv:2402.02716

5 Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of Advances in Neural Information Processing Systems, 2022. 35: 24824–24837

6 Wang X, Wei J, Schuurmans D, et al. Self-consistency improves chain of thought reasoning in language models. 2022. ArXiv:2203.11171

7 Yao S, Zhao J, Yu D, et al. React: synergizing reasoning and acting in language models. 2022. ArXiv:2210.03629

8 Kojima T, Gu S S, Reid M, et al. Large language models are zero-shot reasoners. In: Proceedings of Advances in Neural Information Processing Systems, 2022. 35: 22199–22213

9 Shinn N, Labash B, Gopinath A. Reflexion: an autonomous agent with dynamic memory and self-reflection. 2023. ArXiv:2303.11366

10 Côté M A, Kádár A, Yuan X, et al. Textworld: a learning environment for text-based games. In: Proceedings of the 27th International Conference on Artificial Intelligence, Stockholm, 2019. 41–75

11 Shridhar M, Yuan X, Côté M A, et al. Alfworld: aligning text and embodied environments for interactive learning. 2020. ArXiv:2010.03768

12 Wang R, Jansen P, Côté M A, et al. Scienceworld: is your agent smarter than a 5th grader? 2022. ArXiv:2203.07540

13 Wang L, Xu W, Lan Y, et al. Plan-and-solve prompting: improving zero-shot chain-of-thought reasoning by large language models. 2023. ArXiv:2305.04091

14 Yao S, Yu D, Zhao J, et al. Tree of thoughts: deliberate problem solving with large language models. 2023. ArXiv:2305.10601

15 Besta M, Blach N, Kubicek A, et al. Graph of thoughts: solving elaborate problems with large language models. 2023. ArXiv:2308.09687

16 Liu B, Jiang Y, Zhang X, et al. LLM+ P: empowering large language models with optimal planning proficiency. 2023. ArXiv:2304.11477

17 Qin Y, Hu S, Lin Y, et al. Tool learning with foundation models. ACM Comput Surv, 2025, 57: 1–40

18 Wu C, Yin S, Qi W, et al. Visual ChatGPT: talking, drawing and editing with visual foundation models. 2023. ArXiv:2303.04671

19 Qin Y, Liang S, Ye Y, et al. ToolLLM: facilitating large language models to master 16000+ real-world APIs. 2023. ArXiv:2307.16789

20 Park J S, O'Brien J, Cai C J, et al. Generative agents: interactive simulacra of human behavior. In: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 2023. 1–22

21 Zhang D, Chen L, Zhang S, et al. Large language model is semi-parametric reinforcement learning agent. 2023. ArXiv:2306.07929

22 Wang W, Dong L, Cheng H, et al. Augmenting language models with long-term memory. 2023. ArXiv:2306.07174

23 Wang G, Xie Y, Jiang Y, et al. Voyager: an open-ended embodied agent with large language models. 2023. ArXiv:2305.16291

24 Wang Z, Cai S, Chen G, et al. Describe, explain, plan and select: interactive planning with LLMs enables open-world multi-task agents. In: Proceedings of the 37th Conference on Neural Information Processing Systems, 2023

25 Yang Z, Qi P, Zhang S, et al. Hotpotqa: a dataset for diverse, explainable multi-hop question answering. 2018. ArXiv:1809.09600

26 Thorne J, Vlachos A, Christodoulopoulos C, et al. Fever: a large-scale dataset for fact extraction and verification. 2018. ArXiv:1803.05355

27 Huang J, Chang K C C. Towards reasoning in large language models: a survey. 2022. ArXiv:2212.10403

28 Sun J, Zheng C, Xie E, et al. A survey of reasoning with foundation models. 2023. ArXiv:2312.11562

29 Yang S, Nachum O, Du Y, et al. Foundation models for decision making: problems, methods, and opportunities. 2023. ArXiv:2303.04129

30 Pan S, Luo L, Wang Y, et al. Unifying large language models and knowledge graphs: a roadmap. IEEE Trans Knowl Data Eng, 2024, 36: 3580–3599

31 Wadhwa S, Amir S, Wallace B C. Revisiting relation extraction in the era of large language models. 2023. ArXiv:2305.05003

32 Papaluca A, Krefl D, Rodriguez S M, et al. Zero-and few-shots knowledge graph triplet extraction with large language models. 2023. ArXiv:2312.01954

33 Zeng A, Liu M, Lu R, et al. Agenttuning: enabling generalized agent abilities for LLMs. 2023. ArXiv:2310.12823

34 Xi Z, Ding Y, Chen W, et al. Agentgym: evolving large language model-based agents across diverse environments. 2024. ArXiv:2406.04151

35 Touvron H, Martin L, Stone K, et al. Llama 2: open foundation and fine-tuned chat models. 2023. ArXiv:2307.09288

36 AI@Meta. Llama 3 model card. 2024. https://github.com/meta-llama/llama3/blob/main/MODEL CARD.md

37 Yang A, Yang B, Hui B, et al. Qwen2 technical report. 2024. ArXiv:2407.10671

38 Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models. 2020. ArXiv:2001.08361