An Efficient Context-Dependent Memory Framework for LLM-Centric Agents

Pengyu Gao†
Independent Researcher
piri.gao@outlook.com

Jinming Zhao*†
Qiyuan Lab
zhaojinming@qiyuanlab.com

Xinyue Chen†
Nanjing University of Aeronautics and Astronautics
cxy_nuaa2012@nuaa.edu.cn

Yilin Long
Peking University
yilinlong@stu.pku.edu.cn

Abstract

In human cognitive memory psychology, the context-dependent effect helps retrieve key memory cues essential for recalling relevant knowledge in problem-solving. Inspired by this, we introduce the context-dependent memory framework (CDMem), an efficient architecture miming human memory processes through multistage encoding, context-aware storage, and retrieval strategies for LLM-centric agents. We propose multistage memory encoding strategies for acquiring high-quality multilevel knowledge: expert encoding compresses raw trajectories from a domain-expert perspective, short-term encoding consolidates experiences from current tasks, and long-term encoding reflects insights from past tasks. For memory storage and retrieval, we design a graph-structured, context-dependent indexing mechanism that allows agents to efficiently and accurately recall the most relevant multilevel knowledge tailored to the current task and environmental context. Furthermore, the proposed CDMem framework is an online learning architecture, enabling agents to efficiently learn and update memory while adapting to novel environments and tasks in real-world applications. We conducted extensive experiments on two interactive decision-making benchmarks in the navigation and manipulation domain, ALFWorld and ScienceWorld. Using GPT-4o-mini, our method surpasses state-of-the-art online LLM-centric approaches, achieving success rates of 85.8% and 56.0%, respectively. We hope this work will serve as a valuable reference for the academic and industrial communities in advancing agent-based applications. The codes are available[1].

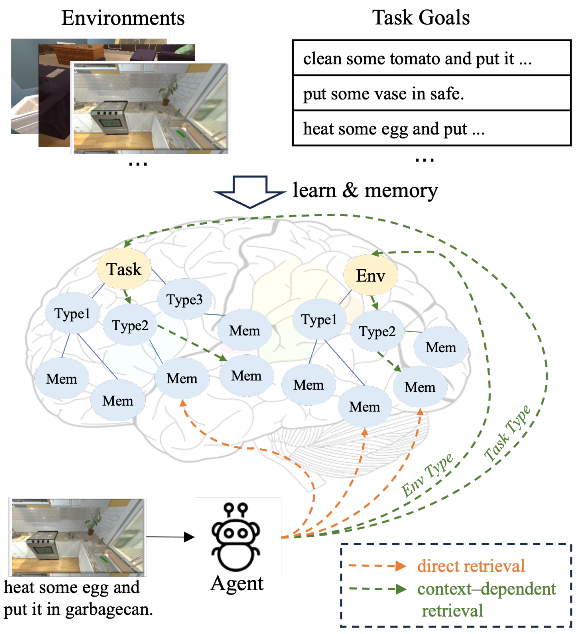


Figure 1: Illustrates the storage and retrieval mechanism in the Context-Dependent Memory (CDMem) framework. Agents can retrieve relevant memories through key cues (task/environment type) or directly access all memories, whereas the former is more efficient and effective. The types of these tasks and environments are predefined by domain experts. "Mem" encompasses knowledge at different levels, including trajectories, task experiences and insights.

1 Introduction

Memory plays a fundamental role in human cognition and brain psychology (Smith and Kosslyn, 2007; Loftus and Loftus, 2019; Xue, 2022), serving as a critical component for learning, storing, and retrieving knowledge, which is equally vital for intelligent agents.

LLM-centric agents have achieved notable success in many decision-making tasks. Some studies have explored the synergy of reasoning traces and specific actions in an interleaved way to improve decision performance (Yao et al., 2023b). (Shinn et al., 2023; Zhao et al., 2024) further introduced a

*Corresponding Author
† Equal contribution.
Pengyu Gao: CDMem implementation, ALFWorld experiments, writing.
Jinming Zhao: CDMem proposal, paper refinement.
Xinyue Chen: ScienceWorld experiments, paper refinement.
[1]https://github.com/piri-gao/CDMem

reflection mechanism, enabling agents to summarize experiences from past trajectories and apply them to subsequent trials or tasks. Recently, some LLM-centric agents have been designed with memory modules to facilitate information storage and retrieval by organizing environmental knowledge into categories or generating state-aware guidelines (Chen et al., 2024; Fu et al., 2024). Memory storage is often inefficient, hindering quick access and selective retrieval, which results in slower and less accurate decisions (Zhong et al., 2024). These methods often struggle to store and retrieve complex information effectively, especially in dynamically changing contexts, preventing agents from leveraging relevant information.

The human brain encodes, stores, and retrieves memories in a context-dependent manner, associating them with specific environments or tasks to quickly identify key cues and activate relevant memory (Smith and Kosslyn, 2007; Xue, 2022). Inspired by this, we propose the context-dependent memory framework (CDMem), an efficient architecture miming human memory processes through multistage encoding, context-aware storage, and retrieval strategies for LLM-centric agents.

We introduce a multistage memory encoding strategy to learn high-quality, multilevel knowledge. First, expert encoding compresses raw trajectories from a domain-expert perspective, mimicking how human experts effectively organize and summarize information after completing a trial. Next, short-term encoding consolidates successes and failures from recent trials within the current task. Finally, long-term encoding integrates insights from past tasks and updates memory indexes to maintain relevance and accuracy. Furthermore, as illustrated in Figure 1, we propose a context-dependent storage indexing mechanism that structures multilevel knowledge within a graph. During retrieval, the agent identifies the current task and environment types and then utilizes the key cues to accurately access relevant exemplars, task experiences, and insights. This precise retrieval process enhances the LLM-centric agent's ability to address and solve the current task effectively. Additionally, the proposed CDMem is an online learning framework that enables agents to efficiently learn and update memory while adapting to novel environments and tasks in real-world applications.

To summarize, our contributions are as follows:

- We propose an efficient online memory paradigm for LLM-centric agents: a context-dependent memory learning, storage, and retrieval framework (CDMem) inspired by the human memory mechanism, particularly suited for developing domain-specific agents in industrial applications.
- We propose an efficient multistage memory learning method, including expert encoding, short-term memory encoding, and long-term memory encoding, to learn multilevel and high-quality knowledge from past tasks;
- We design a context-dependent graph-based indexing that allows agents to efficiently and accurately retrieve the most relevant knowledge through environmental and task-specific cues;
- We conduct extensive experiments on two interactive decision-making benchmarks (ALFWorld and ScienceWorld), We demonstrate that our method outperforms state-of-the-art online LLM-centric memory-based methods, achieving significant performance improvements.

2 Related Work

2.1 LLM for Reasoning and Decision-Making

The introduction of Chain-of-Thought (CoT) (Wei et al., 2022) has significantly enhanced the reasoning capabilities of LLM. Building on this, several works (Kojima et al., 2022; Yao et al., 2023b; Wu et al., 2023) have demonstrated the immense potential of LLM in reasoning and decision-making, surpassing traditional reinforcement learning methods in specific scenarios. Tree-of-Thought (Yao et al., 2023a) and Graph-of-Thought (Besta et al., 2024) further enhanced CoT by extending the linear CoT structure to tree-based and graph-based structures, respectively. Many other works have applied the reasoning and decision-making capabilities of LLM to various domains, including robotics (Ahn et al., 2022; Liang et al., 2023), gaming (Wang et al., 2024b, 2023; Zhu et al., 2023), and game theory (Zhang et al., 2024; Guo et al., 2023). In these complex domains, fully leveraging learned experiences and dynamically forming new experiences based on real-time environmental feedback is crucial for decision-making.

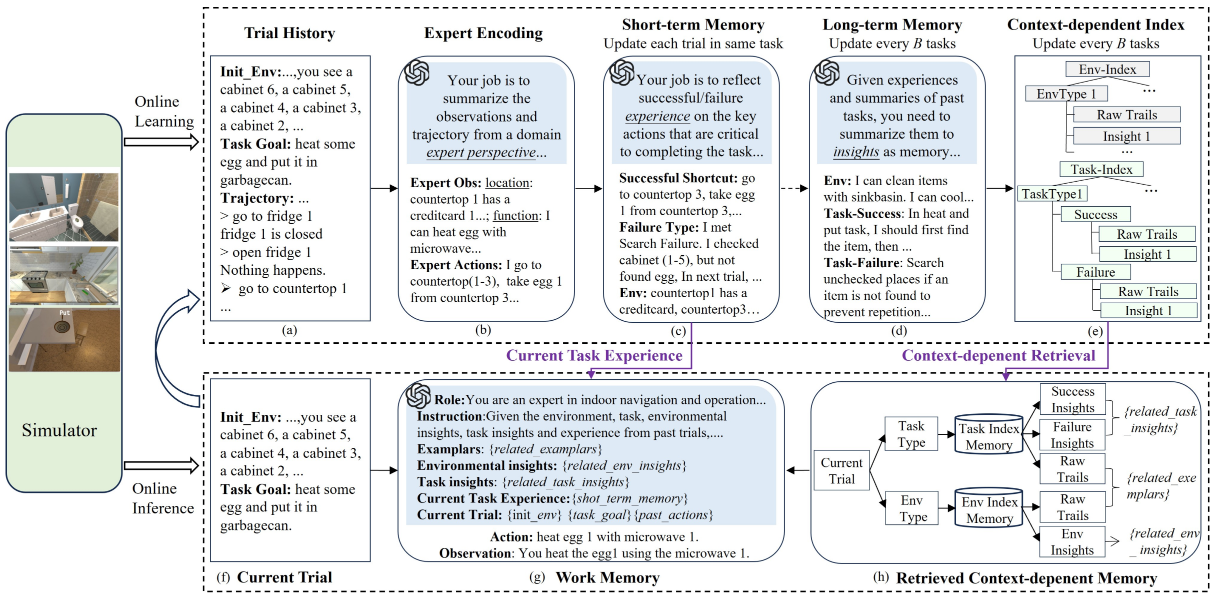

Trial History
Expert Encoding
Short-term Memory
Update each trial in same task
Long-term Memory
Update every B tasks
Context-dependent Index
Update every B tasks
Online Learning
Init_Env:...,you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, ...
Task Goal: heat some egg and put it in garbagecan.
Trajectory: ...
> go to fridge 1
fridge 1 is closed
> open fridge 1
Nothing happens.
➢ go to countertop 1
...
Your job is to summarize the observations and trajectory from a domain expert perspective...
Expert Obs: location: countertop 1 has a creditcard 1...; function: I can heat egg with microwave...
Expert Actions: I go to countertop(1-3), take egg 1 from countertop 3...
Your job is to reflect successful/failure experience on the key actions that are critical to completing the task...
Successful Shortcut: go to countertop 3, take egg 1 from countertop 3,...
Failure Type: I met Search Failure. I checked cabinet (1-5), but not found egg, In next trial, ...
Env: countertop1 has a creditcard, countertop3...
Given experiences and summaries of past tasks, you need to summarize them to insights as memory...
Env: I can clean items with sinkbasin. I can cool...
Task-Success: In heat and put task, I should first find the item, then ...
Task-Failure: Search unchecked places if an item is not found to prevent repetition...
Env-Index
EnvType 1
Raw Trails
Insight 1
Task-Index
TaskType1
Success
Raw Trails
Insight 1
Failure
Raw Trails
Insight 1
(a)
(b)
(c)
(d)
(e)
Current Task Experience
Context-depenent Retrieval
Simulator
Online Inference
Init_Env: ...,you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, ...
Task Goal: heat some egg and put it in garbagecan.
Role:You are an expert in indoor navigation and operation...
Instruction:Given the environment, task, environmental insights, task insights and experience from past trials,....
Examplars: {related_examplars}
Environmental insights: {related_env_insights}
Task insights: {related_task_insights}
Current Task Experience:{shot_term_memory}
Current Trial: {init_env} {task_goal} {past_actions}
Action: heat egg 1 with microwave 1.
Observation: You heat the egg1 using the microwave 1.
Current Trial
Task Type
Env Type
Task Index Memory
Env Index Memory
Success Insights
Failure Insights
Raw Trails
Raw Trails
Env Insights
{related_task _insights}
{related_exe mplars}
{related_env _insights}
(f) Current Trial
(g) Work Memory
(h) Retrieved Context-depenent Memory

Figure 2: **Illustration of the Context-Dependent Memory (CDMem) framework.** (a) The agent interacts with the environment to generate a trial history. (b) The expert encoding module compress the raw trial history and extract expert history including environment, task goal and expert trajectory description. (c) The short-term memory encoding based on the compressed output of (b) to generate successful shortcuts, experiences of defined failure types, and environmental summary. (d) The long-term memory encoding captures cross-task insights, including environmental insights as well as success- and failure-related task insights, based on accumulated short-term memories every B tasks. (e) Organize the knowledge learned from the previous steps into a graph-structured storage index according to task type and environment type. (f) At the inference stage, given a new trial with a description of the environment and task goal. (g) Retrieve relevant knowledge from the context-dependent memory, including exemplars, environmental insights, as well as success- and failure-related task insights. (h) Organize the retrieved knowledge, current trial, and task experiences to form the prompt (similar to working memory in the human brain), and use it to make action decisions through the LLM.

2.2 Memory Storage and Retrieval of Agents

Designing an effective memory mechanism is essential for improving the performance of decision-making agents. MemoryBank (Zhong et al., 2024) proposed a long-term memory mechanism that addresses the lack of long-term memory in LLM by incorporating storage, retrieval, and update mechanisms combined with the Ebbinghaus forgetting curve theory. ChatDB (Hu et al., 2023) introduced databases as symbolic memory for LLM and proposed the Chain-of-Memory (CoM), enhancing the complex reasoning capabilities of LLM. TiM (Liu et al., 2023)improved the performance of LLM in long-term interactions by storing historical thoughts, updating memory through operations such as insertion, forgetting, and merging, and utilizing locality-sensitive hashing for efficient retrieval. Nevertheless, these works all rely on direct retrieval from the entire memory pool without constructing more efficient indexing mechanisms, which leads to inefficient retrieval and insufficient accuracy.

2.3 Memory Self-Learning of Agents

Reflexion(Shinn et al., 2023) converts environmental feedback into textual statements and stores them in memory, allowing the agent to utilize this memory in subsequent trials to improve task performance. Retroformer(Yao et al., 2024) and Reflect-RL(Zhou et al., 2024) further enhanced Reflexion by incorporating reinforcement learning to train specific components, effectively embedding part of the memory into model parameters to improve the agent's reasoning capabilities. In-Memory Learning(Wang et al., 2024a) proposed a framework that constructs memory and enables agent self-improvement through induction, revision, and inference. Expel(Zhao et al., 2024) introduced an offline learning agent that collects experiences through trial-and-error interactions with the environment during the training phase, storing them in an experience pool for later extraction of insights. During the evaluation phase, the agent uses these insights and successful trajectories to assist decision-making. AutoGuide(Fu et al., 2024) extracts a set of state-aware guidelines through offline learning, providing more targeted guidance to the agent based on the current state during testing. Unlike Expel and AutoGuide, CDMem is an online memory-based method that continuously self-improves through real-time memory updates.

3 Method

When humans perform tasks in a specific environment, they use environment- and task-specific cues to retrieve relevant memories efficiently. After completing a task, they organize and consolidate these memories for future use. Inspired by this process, we propose the Context-Dependent Memory (CDMem) framework, which includes three key components: memory encoding, context-dependent memory storage, and context-dependent memory retrieval. This framework efficiently retrieves relevant exemplars, experiences, and insights based on the current environment and task instructions. Memory encoding uses a novel multistage memory learning strategy, while memory storage and retrieval rely on a context-dependent indexing structure. Detailed method descriptions with pseudo code are provided in the Appendix.

3.1 Multistage Memory Encoding

3.1.1 Expert Encoding

Memories formed by domain experts are typically more concise and organized than those of non-experts. This is because expert encoding efficiently groups raw trajectories into knowledge chunks. For instance, a professional chess player will deduce the tactics used in a game, thereby remembering the arrangement of the pieces, whereas a novice would attempt to remember the position of each chess piece from the outset. Inspired by this, we introduce the Expert Encoding Module $\mathcal{M}_{\text{expert}}$ which takes a raw trajectory τ as input and outputs compressed expert observations and actions E_{expert}.

$$E_{\text{expert}} = \mathcal{M}_{\text{expert}}(\tau) \tag{1}$$

Expert observations provide a concise description of the environment, summarizing object locations and their properties within the current environment. Expert actions are well-organized trajectories from an expert's perspective, which omit unnecessary details and consolidate similar actions into a single statement to reduce redundancy.

3.1.2 Short-term Memory Encoding

When an agent repeatedly attempts a task in a specific environment, it reflects on its actions, creating memories tailored to that task and environment, like human short-term memory. To model this process, we introduce the Short-term Memory Encoding Module $\mathcal{M}_{\text{short}}$, which takes raw trajectories and expert encoding as inputs and generates short-term memories as output.

$$E_{\text{short}} = \mathcal{M}_{\text{short}}(\tau, E_{\text{expert}}) \tag{2}$$

This module is similar to the reflection process in Reflexion(Shinn et al., 2023), but with two key improvements inspired by human memory:

(a) Reflection on Successful and Unsuccessful Trajectories Unlike Reflexion, which only reflects on unsuccessful trajectories, our approach separately reflects on both successful and unsuccessful ones. For successful trajectories, the agent reflects on which actions were necessary and which were not, removing unnecessary steps to create a "Successful Shortcut." This represents the shortest path to complete the task and helps the agent focus on essential planning. For unsuccessful trajectories, the agent analyzes the error type, such as planning, search, or operation failures, and then adjusts its plan accordingly.

(b) Environmental Memory. Beyond learning the experiences through reflection, the short-term memory encoding module also learns memories of the current environment via different aspects. When the agent makes attempts in the same or similar environment, the environment memories can help the agent to effective and efficient understanding of the environment.

3.1.3 Long-term Memory Encoding

When an agent performs different tasks in various environments, similarities between these tasks and environments may emerge. These similarities can be summarized into high-level, abstract memories spanning tasks and environments. These memories are recalled not only when the agent encounters the same task or environment but also when it faces similar ones, demonstrating strong generalization and persistence, much like human long-term memory. To capture this, we designed the Long-term Memory Encoding Module $\mathcal{M}_{\text{long}}$, which takes short-term memories as inputs and generates environmental and task insights as output.

$$\begin{aligned} &env\_insights, task\_insights \\ &\quad = \mathcal{M}_{\text{long}}(\tau, E_{\text{short}}) \end{aligned} \tag{3}$$

(a)Environmental Insights. Using ALFWorld and ScienceWorld as examples, environmental insights focus on encoding object properties, such as "a microwave can heat food," rather than summarizing object positions, as in short-term memory. This approach mirrors human memory patterns, where object positions are not easily generalized across environments.

(b)Task Insights. Similar to short-term memory encoding, task insights separately summarize positive and negative examples. For successful memories, the agent combines multiple successful shortcuts to create a general plan for a task category. For unsuccessful memories, the agent consolidates reflections on errors to identify common mistakes and their remedies across tasks.

3.2 Context-Dependent Memory Storage

We propose the Context-Dependent Memory (CDMem) framework, featuring a context-dependent indexing structure that includes both environment-dependent and task-dependent indices. This structure uses environmental and task-specific cues to improve long-term memory retrieval. This subsection outlines the memory storage process.

The context-dependent indexing structure consists of two dictionaries: the Env-Index for indexing environmental long-term memories and the Task-Index for task-related long-term memories. The Task-Index is further divided into two subdictionaries: Success and Failure, which store summarized successful and unsuccessful short-term memories. The dictionary keys represent environment or task descriptions, while the values contain pairs of short-term memories and their corresponding raw trajectories.

When a new short-term memory is created, the corresponding dictionary is updated based on the environment and task. The system searches for matching environment or task descriptions. If a match is found, the memory and its trajectory are added to the list. If no match exists, a new key is created with an empty list, and the memory is added. Once the list reaches a batch size, the Long-term Memory Encoding Module processes the entries into long-term memories, storing environmental insights in the Env-Index and task insights (Success or Failure) in the Task-Index.

3.3 Context-Dependent Memory Retrieval

During inference, the agent retrieves and organizes information based on the current task and environment using the context-dependent indexing structure. To retrieve the corresponding insights and raw trajectories, we propose a prompt-based Index Matching Module $\mathcal{M}_{\text{match}}$, which takes the current task and environment as input and outputs the best-matching environment and task types. The next step is determining which insights and raw trajectories to recall as exemplars.

(a) Retrieval and recall exemplars. The agent retrieves relevant trajectories (CD-exemplars) from both the Env-Index and Task-Index, then prioritizes those in the intersection, where both task and environment match. If more exemplars are needed, the system recalls additional trajectories from the Task-Index or falls back on default exemplars from Reflexion. The Env-Index is not considered at this stage.

(b) Retrieval and re-ranking insights. To filter and rank insights, we propose a non-LLM-based sorting algorithm. It calculates the cosine similarity between each insight and short-term memories in the current environment or task, then sums the scores to prioritize the most relevant insights.

4 Experiments

4.1 Setting

We validated the effectiveness of CDMem and conducted analyses on typical complex interactive reasoning tasks, navigation, and manipulation of situated virtual textual environments: ALFWorld and ScienceWorld. In ALFWorld, agents interact with different rooms to complete household tasks. Following the setting in React(Yao et al., 2023b), we selected 134 environments from ALFWorld as our test set. These 134 environments are composed of 9 rooms and 6 task types. In ScienceWorld, which is similar to ALFWorld, the tasks are more complex. We selected tasks that could be completed within 20 steps to form our test set, which includes a total of 50 tasks.

4.1.1 Baselines

Since this study primarily focuses on how agents generate and utilize memory during online interactions, we choose Reflexion, Expel and AutoGuide as baselines. To ensure a fair comparison, we implemented online versions of Expel and AutoGuide, referred to as "Expel-Online" and "AutoGuide-

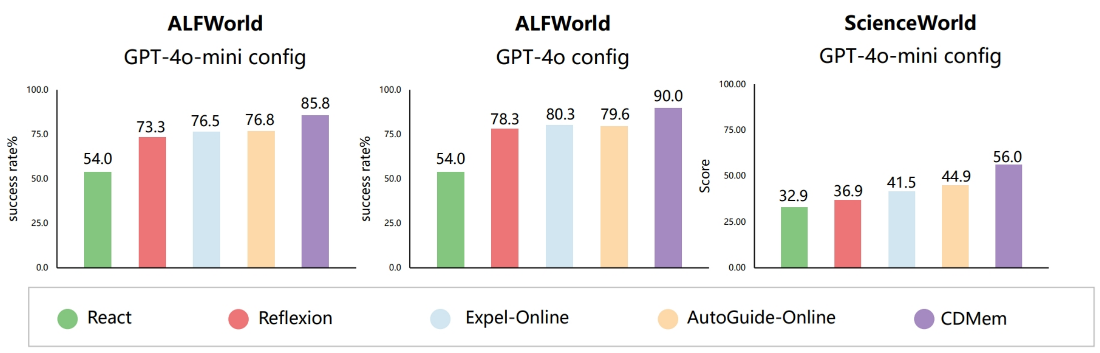


Figure 3: Main Results. Average task success rates on ALFWorld with two model configurations and average scores on ScienceWorld with the base model configuration.

Method	Success Rate%
CDMem w/o expert	72.3
CDMem w/o task-encoding	84.3
CDMem w/o env-encoding	74.3
CDMem w/o CD-Exemplars	80.6
CDMem	85.8

Table 1: Ablation Study on ALFWorld.(a) CDMem w/o expert: CDMem without expert encoding;(b) CDMem w/o task-encoding: Removal of task insights from the long-term encoding; (c) CDMem w/o env-encoding: Removal of environmental insights from the long-term memory encoding; (d)CDMem w/o CD-Exemplars: Instead of using context-dependent memory for exemplars, using fixed exemplars in Reflexion(Shinn et al., 2023).

Method	Score
CDMem w/o task-encoding	40.9
CDMem w/o env-encoding	54.7
CDMem	56.0

Table 2: Ablation Study on ScienceWorld

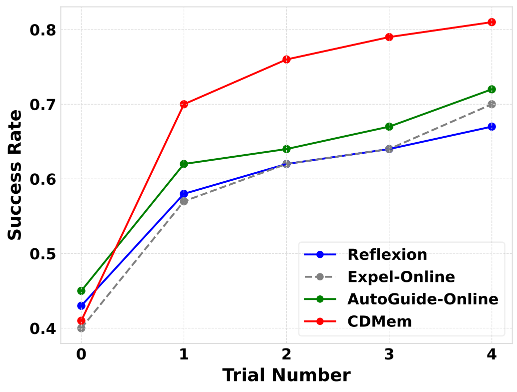


Figure 4: The curve of the relationship between success rate and the number of trials.

Online". Furthermore, we also selected the React algorithm, which does not involve memory, as a baseline to reflect the model's performance without using any memory methods.

4.1.2 Implementation

We conducted 5 trials in each environment of ALFWorld and ScienceWorld, with a maximum of 20 interaction steps per trial and environment. All methods use two exemplars. All experiments were run three times, and the experimental results were averaged. We evaluated our model with two configurations:(a) GPT-4o-mini: All components use GPT-4o-mini. (b) GPT-4o: Memory-related components use GPT-4o, while other components use GPT-4o-mini.

4.2 Main Results

In both ALFWorld and ScienceWorld, CDMem achieved significant improvements over the baselines(see Figure 3).With Configuration 1, CDMem achieved a success rate of 85.8%, a 9% improvement over the AutoGuide-Online. Similarly, in ScienceWorld, CDMem scored 56.02, exceeding AutoGuide-Online's score of 44.84 by more than 10 points. With Configuration 2, CDMem's success rate on ALFWorld reached 90.0%, nearly a 10% increase over Expel-Online's 80.3%.

4.3 Ablation Studies

We verified the modules of CDMem contribute to performance improvement on ALFWorld shown in Table 1 and on ScienceWorld shown in Table 2. It can be seen that each component of CDMem plays an important role.

(a)Role of Expert Encoding Expert encoding compresses the raw trajectory and allows larger batch sizes during updates. Second, expert encoding improves the accuracy of successful shortcut summarization. Rather than directly extracting shortcuts from successful trajectories, the agent first summarizes expert actions. This helps the agent focus only on identifying unnecessary actions when summarizing successful shortcuts.

(b)Role of Short-term Memory A well-designed reflection mechanism enables significant improvement in an agent's performance on subsequent trials of the same task. In our tests on ALFWorld (see Figure 4), CDMem showed a more substantial improvement between trial 0 and trial 1 compared to other methods.

(c)Role of Environmental Insights In ALFWorld, removing expert encoding and environmental insights had the largest impact and reduced about 10-points of SR. However, in the ScienceWorld environment, removing environmental insights had only a 2-point impact. Moreover, we found that environmental insights significantly reduce hallucinations in the agent's behavior. For example, in ALFWorld, these insights provide accurate contextual information for item (such as microwave) usage.

(d)Role of Task Insights Task insights are similar to the status guidelines in AutoGuide. Compared to Expel, which uses all available insights, task insights are more focused and relevant to the current task, offering more accurate guidance. Similar to the role of environmental insights, task insights are also essential in mitigating hallucinations in the agent's behavior. For example, in ALFWorld, successful task insights outline the correct sequence of actions for a task, helping the agent avoid performing actions that fall outside the planned steps due to hallucinations.

(e)Role of CD-Exemplar Although the fixed exemplars provided by Reflexion are also task-dependent, CD-Exemplars represent the actual interaction trajectories of the agent with the environment, making them more valuable as references for the agent. As a result, this led to an improvement of nearly 5% in ALFWorld.

4.4 Computational Cost

We compared the computational cost of CDMem with Reflexion. Specifically, using the GPT-4o-mini configuration, we conducted five trials across 20 randomly selected environments from ALFWorld to compare the computational cost shown in Table 3, which includes the number of API calls per individual sample, the total number of API calls for the selected dataset, and the corresponding monetary cost for processing the dataset.

Computational Cost	Reflexion	CDMem
API Calls per Sample	2	4
Total API Calls (Dataset)	781	1155
Monetary Cost (Dataset)	\$0.33	\$0.51

Table 3: Comparison of Computational Costs

5 Conclusion

We introduce CDMem, an efficient online memory framework for LLM-centric agents inspired by human memory mechanisms and designed for domain-specific industrial applications. Our approach incorporates a multi-stage memory learning method—expert encoding, short-term memory encoding, and long-term memory encoding—to effectively capture and organize knowledge from past tasks. We also introduce a context-dependent graph-based indexing structure that allows agents to retrieve relevant knowledge efficiently. We demonstrate that CDMem significantly outperforms state-of-the-art methods through extensive experiments, achieving notable performance improvements.

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, and Karol Hausman et al. 2022. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning*, volume 205, pages 287–318.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 17682–17690.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. 2024. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. In *Advances in Neural Information Processing Systems*, volume 37, pages 589–631.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, pages 119919–119948.

Jiaxian Guo, Bo Yang, Paul Yoo, Bill Yuchen Lin, Yusuke Iwasawa, and Yutaka Matsuo. 2023. Suspicion-agent: Playing imperfect information games with theory of mind aware GPT-4. *arXiv preprint arxiv:2309.17277*.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as policies: Language model programs for embodied control. In *International Conference on Robotics and Automation*, pages 9493–9500.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. 2023. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. *arXiv preprint arXiv:2311.08719*.

Geoffrey R Loftus and Elizabeth F Loftus. 2019. *Human memory: The processing of information*. Psychology Press.

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*, volume 36, pages 8634–8652.

Edward E Smith and Stephen Michael Kosslyn. 2007. *Cognitive psychology: Mind and brain*. Pearson-/Prentice Hall.

Bo Wang, Tianxiang Sun, Hang Yan, Siyin Wang, Qingyuan Cheng, and Xipeng Qiu. 2024a. In-memory learning: A declarative learning framework for large language models. *arXiv preprint arxiv:2403.02757*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024b. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024.

Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: Interactive planning with LLMs enables open-world multi-task agents. In *Advances in Neural Information Processing Systems*.

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*, volume 35, pages 24824–24837.

Yue Wu, So Yeon Min, Shrimai Prabhumoye, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. 2023. Spring: Studying the paper and reasoning to play games. In *Advances in Neural Information Processing Systems*.

Gui Xue. 2022. From remembering to reconstruction: The transformative neural representation of episodic memory. *Progress in Neurobiology*, 219:102351.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Weiran Yao, Shelby Heinecke, Juan Carlos Niebles, Zhiwei Liu, Yihao Feng, Le Xue, Rithesh Murthy, Zeyuan Chen, Jianguo Zhang, and Devansh Arpit et al. 2024. Retroformer: Retrospective large language agents with policy gradient optimization. In *International Conference on Learning Representations*.

Wenqi Zhang, Ke Tang, Hai Wu, Mengna Wang, Yongliang Shen, Guiyang Hou, Zeqi Tan, Peng Li, Yueting Zhuang, and Weiming Lu. 2024. Agent-pro: Learning to evolve via policy-level reflection and optimization. In *Proceedings of the Annual Meeting of the Association for Computational*, pages 5348–5375.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

Runlong Zhou, Simon S. Du, and Beibin Li. 2024. Reflect-rl: Two-player online RL fine-tuning for lms. In *Proceedings of the Annual Meeting of the Association for Computational*, pages 995–1015.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Yu Qiao, Zhaoxiang Zhang, and Jifeng Dai. 2023. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2309.17277*.

A Appendix

A.1 Environment Details

We conduct experiments on the CDMem algorithm using two virtual textual environments: ALFWorld and ScienceWorld. Since the test set for ALFWorld is the same as those used in Reflexion and Expel, this section primarily introduces the construction of the ScienceWorld test set. The tasks in ScienceWorld are divided into 30 task types, each containing multiple variants. We select 10 types of tasks where the average length of the oracle agent's trajectories is **less than 20**. For each Task Type, we choose the top 5 variants, resulting in a total of 50 environments, which are shown in Table 4

A.2 Implementation Details

In Table 5, we provide the specific version of the models used in the experiments.

Model	Version
GPT-4o	gpt-4o-2024-05-13
GPT-4o-mini	gpt-4o-mini-2024-07-18
Embedding model	text-embedding-3-small

Table 5: Version numbers of the models.

A.3 Prompt Templates

We present our prompt templates for different modules in Figures 5-11.

A.4 Additional Experiments and Analyses

A.4.1 Role of Expert Encoding

We analyze the role of expert encoding and find that applying expert encoding before short-term memory encoding results in more accurate summaries of successful shortcuts than directly using raw trajectories. We believe that the action sequence summarized by the expert encoding provides a reference for short-term memory encoding, reducing the cognitive load on the LLM when identifying redundant actions. This appearance is similar to the effectiveness of the "think step by step" approach. Here is an example of the appearance in Figure 12.

A.4.2 Effect of batch size

CDMem supports a larger batch size in the insights extraction phase than Expel and AutoGuide. We conducted batch size experiments on ALFWorld, and the results indicate that even with a batch size as large as 11, insights can still be effectively extracted. The experimental results are shown in the Table 6

Batch size	Success Rate%
1	85.8
3	84.6
5	86.4
7	86.2
9	83.6
11	84.3

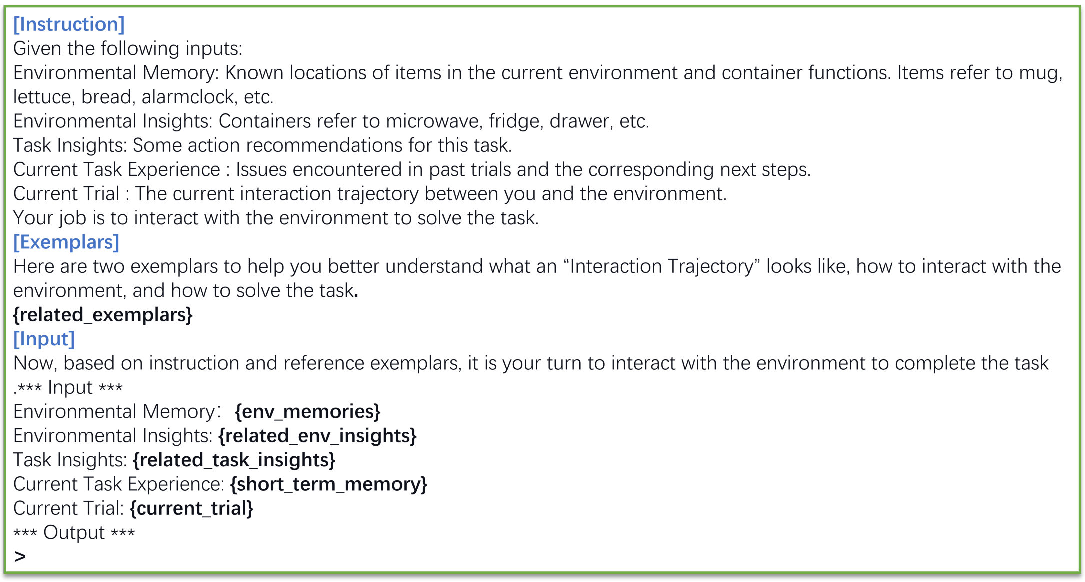
Table 6: Batch size experimental results on ALFWorld

A.5 Pseudo code of methods

The pseudo code for the multistage memory encoding and storage, memory retrieval, and results rerank are shown in Algorithm 1-3

Task Type	Topic	Name	***Lens**	**# Variations**	**Chosen**
1-1	Matter	Changes of State (Boiling)	107.7	30	
1-2	Matter	Changes of State (Melting)	78.6	30	
1-3	Matter	Changes of State (Freezing)	88.9	30	
1-4	Matter	Changes of State (Any)	75.2	30	
2-1	Measurement	Use Thermometer	21.4	540	
2-2	Measurement	Measuring Boiling Point (known)	35.2	436	
2-3	Measurement	Measuring Boiling Point (unknown)	65	300	
3-1	Electricity	Create a circuit	13.6	20	✓
3-2	Electricity	Renewable vs Non-renewable Energy	20.8	20	
3-3	Electricity	Test Conductivity (known)	25.6	900	
3-4	Electricity	Test Conductivity (unknown)	29	600	
4-1	Classification	Find a living thing	14.6	300	✓
4-2	Classification	Find a non-living thing	8.8	300	✓
4-3	Classification	Find a plant	12.6	300	✓
4-4	Classification	Find an animal	14.6	300	✓
5-1	Biology	Grow a plant	69.5	126	
5-2	Biology	Grow a fruit	79.6	126	
6-1	Chemistry	Mixing (generic)	33.6	32	
6-2	Chemistry	Mixing (generic)	15.1	32	✓
6-3	Chemistry	Mixing (generic)	23	36	
7-1	Biology	Identify longest-lived animal	7	125	✓
7-2	Biology	Identify shortest-lived animal	7	125	✓
7-3	Biology	Identify longest-then-shortest-lived animal	8	125	✓
8-1	Biology	Identify life stages (plant)	40	14	
8-2	Biology	Identify life stages (animal)	16.3	10	✓
9-1	Forces	Inclined Planes (determine angle)	97	168	
9-2	Forces	Friction (known surfaces)	84.9	1386	
9-3	Forces	Friction (unknown surfaces)	123.1	162	
10-1	Biology	Mendelian Genetics (known plants)	130.1	120	
10-2	Biology	Mendelian Genetics (unknown plants)	132.1	480	

Table 4: **Chosen Environments of ScienceWorld benchmark.** **Lens* is the average length of the oracle agent's trajectories.



[Instruction]
Given the following inputs:
Environmental Memory: Known locations of items in the current environment and container functions. Items refer to mug, lettuce, bread, alarmclock, etc.
Environmental Insights: Containers refer to microwave, fridge, drawer, etc.
Task Insights: Some action recommendations for this task.
Current Task Experience : Issues encountered in past trials and the corresponding next steps.
Current Trial : The current interaction trajectory between you and the environment.
Your job is to interact with the environment to solve the task.
[Exemplars]
Here are two exemplars to help you better understand what an "Interaction Trajectory" looks like, how to interact with the environment, and how to solve the task.
{related_exemplars}
[Input]
Now, based on instruction and reference exemplars, it is your turn to interact with the environment to complete the task
.*** Input ***
Environmental Memory： **{env_memories}**
Environmental Insights: **{related_env_insights}**
Task Insights: **{related_task_insights}**
Current Task Experience: **{short_term_memory}**
Current Trial: **{current_trial}**
*** Output ***
>

Figure 5: Our prompt template for inference.

[Instruction]
Given the interaction trajectory of current trial , Your job is to summarize the trajectory from a domain expert perspective that includes the following parts:
1. Expert Observations: (1) the location where items (such as mug, lettuce, bread, alarm clock) are placed, for example, "drawer 1 has a mug, shelf 2 has an alarm clock";(2) the functions of some containers (such as drawer, shelf, sinkbasin, fridge). For example, "I can clean lettuce with a sink basin; I can cool a mug with a fridge." If no container's function can be summarized, output "None."
2. Expert Actions: a brief summarization of the action trajectories following the original execution order and ignoring the thought process inside. If adjacent actions are similar, some simplification can be made.
[Exemplars]
There are three exemplars to help you to understand what "Expert Observations" and "Expert Actions" are and how to generate them:
{fewshots}
[Input]
Based on instruction and reference exemplars, it is your turn to summarize the trajectory into an expert memory, including " Expert Observations" and "Expert Actions."
*** Input ***
Current Trial: **{current_trial}**
*** Output ***
Expert Observations:
Expert Actions:

Figure 6: Our prompt template for expert encoding.

[Instruction]
You have completed the task in this trial. Now, given the interactive trajectory of current trial , expert actions(summary of interactive trajectory), and memory you make in past trials, your job is to generate a successful shortcut on the key actions that are critical to completing the task, which means that eliminating any of these actions would affect the completion of the task.
[Exemplars]
There are two exemplars to help you better understand what "successful shortcuts" are and the memory you should build.
{fewshots}
[Input]
Based on instruction and reference exemplars, it is your turn to build memory on the successful shortcut.
*** Input ***
Current Trial : **{current_trial}**
Expert Actions: **{expert_actions}**
Past Memories :**{past_memories}**
*** Output ***
Successful Shortcut:

Figure 7: Our prompt template for short-term memory encoding of successful trajectory.

[Instruction]

You were unsuccessful in completing the task in this trial. Now, Your job is to build memory in two aspects:

1. Given the current trial's item location information(Items refer to mug, lettuce, bread, alarm clock, etc.) and past environmental memory (summary of item locations in this environment), summarize them to form new environmental memory and output it.
2. Given interactive trajectory of current trial, environmental memory, expert actions(summary of action trajectories), and past reflections(reflections you made in past trials), you must first consider what types of failure you meet and output corresponding reflections.There are three types of failure:

Planning Failure: The task planning has issues, such as missing steps or misunderstandings. Output the reflection of current planning issues and the correct plan.

Search Failure: Continuously searching for an item but unable to find it. Output the item's location already searched and the reflection of the future search plan. For example, if you tried A and B but forgot C, devise a plan to achieve C with environment-specific actions.

Operation Failure: The expected feedback was not received after acting, such as returning with "nothing happens," which means the current observation doesn't match the current action. For example, attempting to take something from cabinet 1 while at the location of cabinet 4 and then returning "nothing happens." Output the reflection of the failed reason and correct actions.

[Exemplars]

There are three exemplars to help you better understand the memory you should build.

{fewshots}

[Input]

Based on instruction and reference exemplars, it is your turn to build memory.

*** Input ***

Current Trial : **{current_trial}**

Current Trial Item Location: **{cur_item_location}**

Environmental Memory: **{enviromental_memory}**

Expert Actions: **{expert_actions}**

Past Failure Memory:**{past_failure_memory**}

*** Output ***

Environmental Memory:

Failure Memory:

Figure 8: Our prompt template for short-term memory encoding of failure trajectory.

[Instruction]

Given multiple experiences of task names, corresponding successful shortcut(key actions which are critical to completing the task) and insights you made in past trials, you need to summarize these experiences and past insights as task insights containing the general planning such as "I should first find the item, then heat it with microwave, and put it in/on container at last. "

[Exemplars]

There are two exemplars to help you better understand the insights you should summarize.

{task_fewshots}

[Input]

Based on instruction and reference exemplars, it is your turn to make a summary.

*** Input ***

Experiences:**{experiences}**

Past Insights:**{task_insights}**

*** Output ***

Figure 9: Our prompt template for long-term memory encoding of successful task insights.

[Instruction]

Given multiple experiences containing task names, corresponding failure memory(type of failure, description of the failure situation, and a plan), and insights you made in past trials, you need to summarize these experiences and past insights as task insights containing the failure situations and corresponding plans.

[Exemplars]

There are two exemplars to help you better understand the insights you should summarize.

{task_fewshots}

[Input]

Based on instruction and reference exemplars, it is your turn to make a summary.

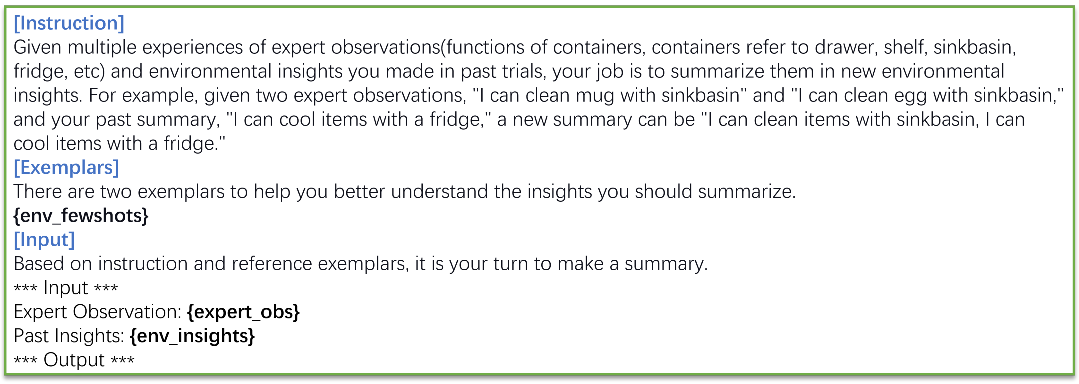
*** Input ***

Experiences**:{experiences}**

Past Insights**:{task_insights}**

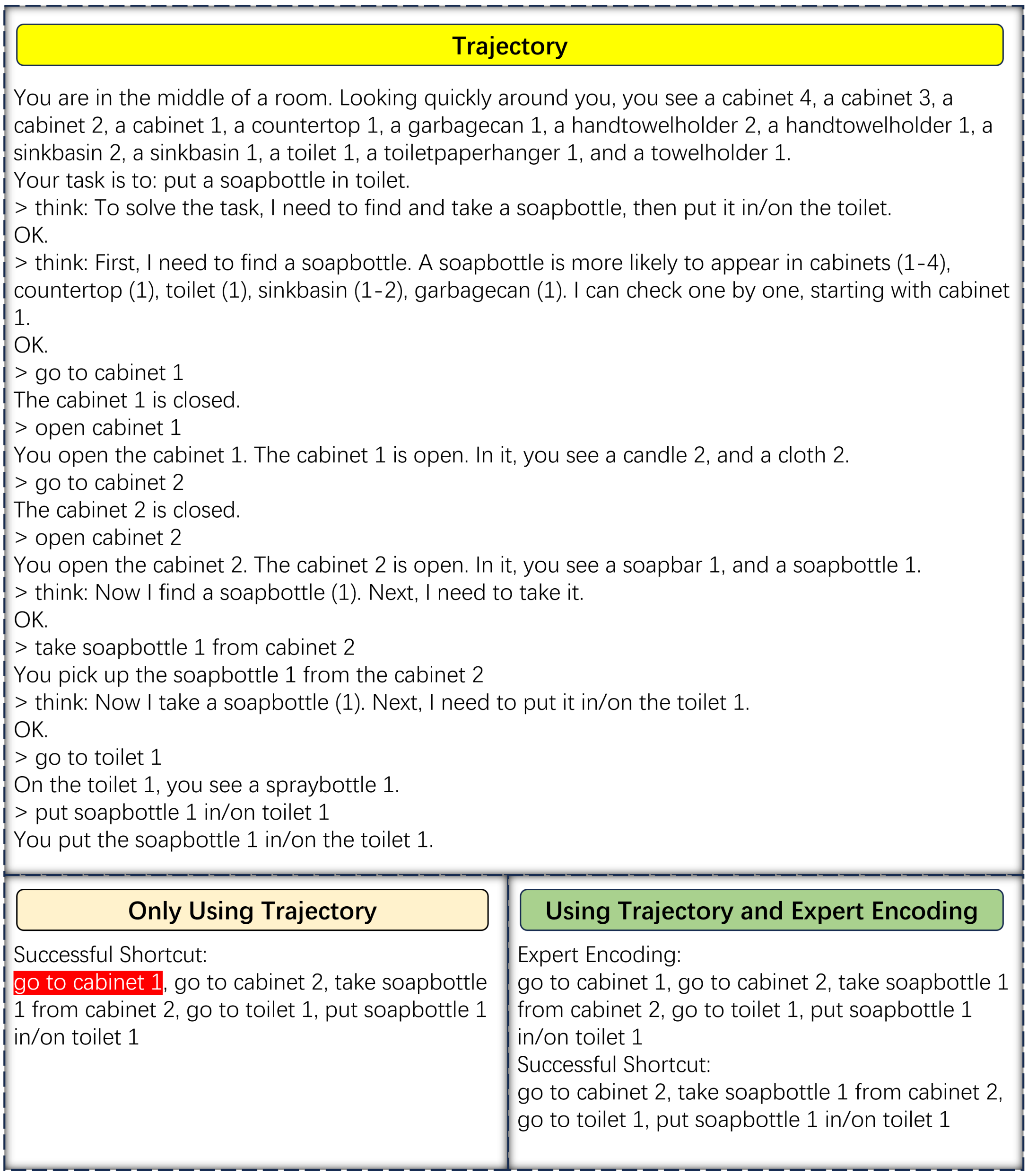
*** Output ***

Figure 10: Our prompt template for long-term memory encoding of failure task insights.



[Instruction]
Given multiple experiences of expert observations(functions of containers, containers refer to drawer, shelf, sinkbasin, fridge, etc) and environmental insights you made in past trials, your job is to summarize them in new environmental insights. For example, given two expert observations, "I can clean mug with sinkbasin" and "I can clean egg with sinkbasin," and your past summary, "I can cool items with a fridge," a new summary can be "I can clean items with sinkbasin, I can cool items with a fridge."
[Exemplars]
There are two exemplars to help you better understand the insights you should summarize.
{env_fewshots}
[Input]
Based on instruction and reference exemplars, it is your turn to make a summary.
*** Input ***
Expert Observation: **{expert_obs}**
Past Insights: **{env_insights}**
*** Output ***

Figure 11: Our prompt template for long-term memory encoding of environmental insights.



Trajectory

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 a soapbottle in toilet.
> think: To solve the task, I need to find and take a soapbottle, then put it in/on the toilet.
OK.
> think: First, I need to find a soapbottle. A soapbottle is more likely to appear in cabinets (1-4), countertop (1), toilet (1), sinkbasin (1-2), garbagecan (1). I can check one by one, starting with cabinet 1.
OK.
> go to cabinet 1
The cabinet 1 is closed.
> open cabinet 1
You open the cabinet 1. The cabinet 1 is open. In it, you see a candle 2, and a cloth 2.
> go to cabinet 2
The cabinet 2 is closed.
> open cabinet 2
You open the cabinet 2. The cabinet 2 is open. In it, you see a soapbar 1, and a soapbottle 1.
> think: Now I find a soapbottle (1). Next, I need to take it.
OK.
> take soapbottle 1 from cabinet 2
You pick up the soapbottle 1 from the cabinet 2
> think: Now I take a soapbottle (1). Next, I need to put it in/on the toilet 1.
OK.
> go to toilet 1
On the toilet 1, you see a spraybottle 1.
> put soapbottle 1 in/on toilet 1
You put the soapbottle 1 in/on the toilet 1.

Only Using Trajectory

Successful Shortcut:
go to cabinet 1, go to cabinet 2, take soapbottle 1 from cabinet 2, go to toilet 1, put soapbottle 1 in/on toilet 1

Using Trajectory and Expert Encoding

Expert Encoding:
go to cabinet 1, go to cabinet 2, take soapbottle 1 from cabinet 2, go to toilet 1, put soapbottle 1 in/on toilet 1
Successful Shortcut:
go to cabinet 2, take soapbottle 1 from cabinet 2, go to toilet 1, put soapbottle 1 in/on toilet 1

Figure 12: Role of Expert Encoding.The left column summarizes the successful shortcuts using only the raw trajectory, while the right column utilizes both the raw trajectory and expert encoding.

Algorithm 1 Multistage Encoding and Memory Storage

Input: Task List $tasks$, Number of tasks K, Maximum number of trials N, Expert encoding module $\mathcal{M}_{\text{expert}}$, Short-term memory encoding module $\mathcal{M}_{\text{short}}$, Long-term memory encoding module $\mathcal{M}_{\text{long}}$, Index matching module $\mathcal{M}_{\text{match}}$, environment dictionary $Env\text{-}Index$, Task dictionary $Task\text{-}Index$, Update Batch Size bs

Output:

Updated $Env\text{-}Index$

Updated $Task\text{-}Index$

```
1: for $i = 1$ to $N$ do
2:     for $j = 1$ to $K$ do
3:         $\tau^{i,j}$ = $Interact\_with\_environment()$
4:         $E^{i,j}_{\text{expert}}$ = $\mathcal{M}_{\text{expert}}(\tau^{i,j})$
5:         $E^{i}_{\text{short}}$ = $\mathcal{M}_{\text{short}}(\tau^{i,j}, E^{i-1}_{\text{short}})$
6:         $task\_type$, $env\_type$ = $\mathcal{M}_{\text{match}}(tasks[\text{j}])$
7:         $Task\text{-}Index[task\_type]$[trajs].add($\tau^{i,j}$, $E^{i}_{\text{short}}$)
8:         $Env\text{-}Index[env\_type]$[trajs].add($\tau^{i,j}$, $E^{i}_{\text{short}}$)
9:         if $len(Task\text{-}Index[task\_type]$ % $bs == 0$ then
10:            $task\_insights^{\text{old}}$
                   = $Task\text{-}Index[task\_type]$[insights]
11:            $task\_insights^{\text{new}}$
                   = $\mathcal{M}_{\text{long}}(E^{\text{batch}}_{\text{short}}, task\_insights^{\text{old}})$
12:            $Task\text{-}Index[task\_type]$[insights]
                   = $task\_insights^{\text{new}}$
13:        end if
14:        if $len(Env\text{-}Index[env\_type])$ % $bs == 0$ then
15:            $env\_insights^{\text{old}}$
                   = $Env\text{-}Index[env\_type]$[insights]
16:            $env\_insights^{\text{new}}$
                   = $\mathcal{M}_{\text{long}}(E^{\text{batch}}_{\text{short}}, env\_insights^{\text{old}})$
17:            $Env\text{-}Index[env\_type]$[insights]
                   = $env\_insights^{\text{new}}$
18:        end if
19:    end for
20: end for
```

Algorithm 2 Memory Retrieval

Input: Task $task$, Index matching module $\mathcal{M}_{\text{match}}$, environment dictionary $Env\text{-}Index$, Task dictionary $Task\text{-}Index$, Number of exemplars needed M, Number of insights needed $2L$, Default exemplar list $default\_exemplars$, Rerank algorithm $Rerank$

Output:
environmental insights $env\_insights$,
Task insights $task\_insights$,
exemplars $CD\_exemplars$

```
1:  $CD\_exemplars = \emptyset$
2:  $task\_type$, $env\_type$ = $\mathcal{M}_{\text{match}}(task)$
3:  $similar\_task\_trajs$, $similar\_task\_short\_memories$
        = $Task\text{-}Index[task\_type]$[trajs]
4:  $similar\_env\_trajs$, $similar\_env\_short\_memories$
        = $Task\text{-}Index[env\_type]$[trajs]
5:  $task\_insights = Task\text{-}Index[task\_type]$
6:  $env\_insights = Env\text{-}Index[env\_type]$
7:  $task\_insights = Rerank(task\_insights,$
                        $similar\_task\_short\_memories$ )
8:  $env\_insights = Rerank(env\_insights,$
                        $similar\_env\_short\_memories$ )
9:  $intersection$
        = $similar\_task\_trajs \cap similar\_env\_trajs$
10: Add exemplars to $CD\_exemplars$ in order of priority:
    $intersection$, $similar\_task\_trajs$,
    $default\_exemplars$
11: **return**
    $task\_insights[:L]$, $env\_insights[:L]$,
    $CD\_exemplars$
```

Algorithm 3 Rerank Algorithm

Input:
Insights $sorted\_insights$
Short Memories $short\_memories$

Output:
Sorted insights $insights$

```
1: **for** each $insight$ in $insights$ **do**
2:     $similarity\_scores$
           = Faiss($short\_memories$, $insight$)
3:     $ranking\_weight$ = sum($similarity\_scores$)
4: **end for**
5: Sort $insight$ in descending order according to their respective the $ranking\_weight$ as
   $sorted\_insights$
6: **return** $sorted\_insights$
```