Mem^p : Exploring Agent Procedural Memory

Mem^p ： 探索智能体的程序性记忆

Runnan Fang [•♈*] , Yuan Liang [•*] , Xiaobin Wang [◦] , Jialong Wu [◦] , Shuofei Qiao [♠♡] , Pengjun Xie [♡] , Fei Huang [♡] , Huajun Chen [♠,] , Ningyu Zhang [♠†]
方润南 [•♈*] ， 梁源 [•*] ， 王晓彬 [◦] ， 吴嘉龙 [◦] ， 乔硕飞 [♠♡] ， 谢鹏军 [♡] ， 黄飞 [♡] ， 陈华军 [♠,] ， 张宁宇 [♠†]

Azhejiang University [♡] Alibaba Group
浙江大学 [♡] 阿里巴巴集团

{rolnan,zhangningyu}@zju.edu.cn
{rolnan,zhangningyu}@zju.edu.cn

1 Abstract

2 摘要

Large Language Models (LLMs) based agents excel at diverse tasks, yet they suffer from brittle procedural memory that is manually engineered or entangled in static parameters. In this work, we investigate strategies to endow agents with a learnable, updatable, and lifelong procedural memory. We propose Mem^p that distills past agent trajectories into both fine-grained, step-by-step instructions and higher-level, script-like abstractions, and explore the impact of different strategies for Build, Retrieval, and Update of procedural memory. Coupled with a dynamic regimen that continuously updates, corrects, and deprecates its contents, this repository evolves in lockstep with new experience. Empirical evaluation on TravelPlanner and ALFWorld shows that as the memory repository is refined, agents achieve steadily higher success rates and greater efficiency on analogous tasks. Moreover, procedural memory built from a stronger model retains its value: migrating the procedural memory to a weaker model yields substantial performance gains.
基于大型语言模型（LLMs）的智能体在多样化任务中表现出色，但其程序性记忆通常是手工设计的或纠缠于静态参数中，因而脆弱。在本工作中，我们研究了赋予智能体可学习、可更新且终身持续的程序性记忆的策略。我们提出Mem^p，将过去的智能体轨迹提炼为细粒度的逐步指令和更高层次的脚本式抽象，并探索了程序性记忆的构建、检索和更新的不同策略。结合动态机制，持续更新、纠正和废弃记忆内容，该记忆库与新经验同步演进。在TravelPlanner和ALFWorld上的实证评估表明，随着记忆库的不断完善，智能体在类似任务上的成功率和效率稳步提升。此外，由更强模型构建的程序性记忆具有持久价值：将其迁移到较弱模型上能显著提升性能。

3 Introduction

4 引言

As large language models (LLMs) grow ever more powerful, LLM-based agents augmented by their own reasoning and external tools are taking on increasingly sophisticated works (Zhao et al. 2023; Wang et al. 2024a; Xi et al. 2025; Qiao et al. 2023). No longer mere assistants, these agents now trawl the web for elusive insights and weave them into comprehensive, publication ready reports, like Deep Research (OpenAI 2025; x.ai 2025) and Web-Dancer (Wu et al. 2025a). Moreover, they can handle complex data analyses (Lan et al. 2025; Ifargan et al. 2025; Ou et al. 2025), navigate multi-step GUI workflows (Luo et al. 2025; Qin et al. 2025), and sustain long-horizon, tool-rich interactions (Yao et al. 2025; Barres et al. 2025; Chen et al. 2025; Fang et al. 2025; Gur et al. 2023) with precision. Yet executing such intricate, long-horizon tasks demands dozens of steps and protracted runtimes. Along the way, unpredictable external events—network glitches, UI changes, shifting data schemas—can derail the entire process. Restarting from scratch every time is a punishing ordeal for present-day agents. Beneath their surface diversity,

many complex tasks share deep structural commonalities
随着大型语言模型（LLMs）能力日益增强，基于LLM并辅以自身推理和外部工具的智能体正承担起越来越复杂的工作（Zhao等，2023；Wang等，2024a；Xi等，2025；Qiao等，2023）。这些智能体不再是简单的助手，而是能够在网络中搜寻难觅的洞见，并将其编织成全面且可发表的报告，如Deep Research（OpenAI，2025；x.ai，2025）和Web-Dancer（Wu等，2025a）。此外，它们能处理复杂数据分析（Lan等，2025；Ifargan等，2025；Ou等，2025）、导航多步骤图形用户界面工作流（Luo等，2025；Qin等，2025），并精准维持长时程、多工具交互（Yao等，2025；Barres等，2025；Chen等，2025；Fang等，2025；Gur等，2023）。然而，执行如此复杂且长时程的任务需要数十步和较长运行时间。过程中，不可预见的外部事件——网络故障、界面变更、数据模式变化——可能导致整个流程中断。每次从头开始对现有智能体来说都是极其艰难的考验。在表面多样性之下，许多复杂任务共享深层结构共性

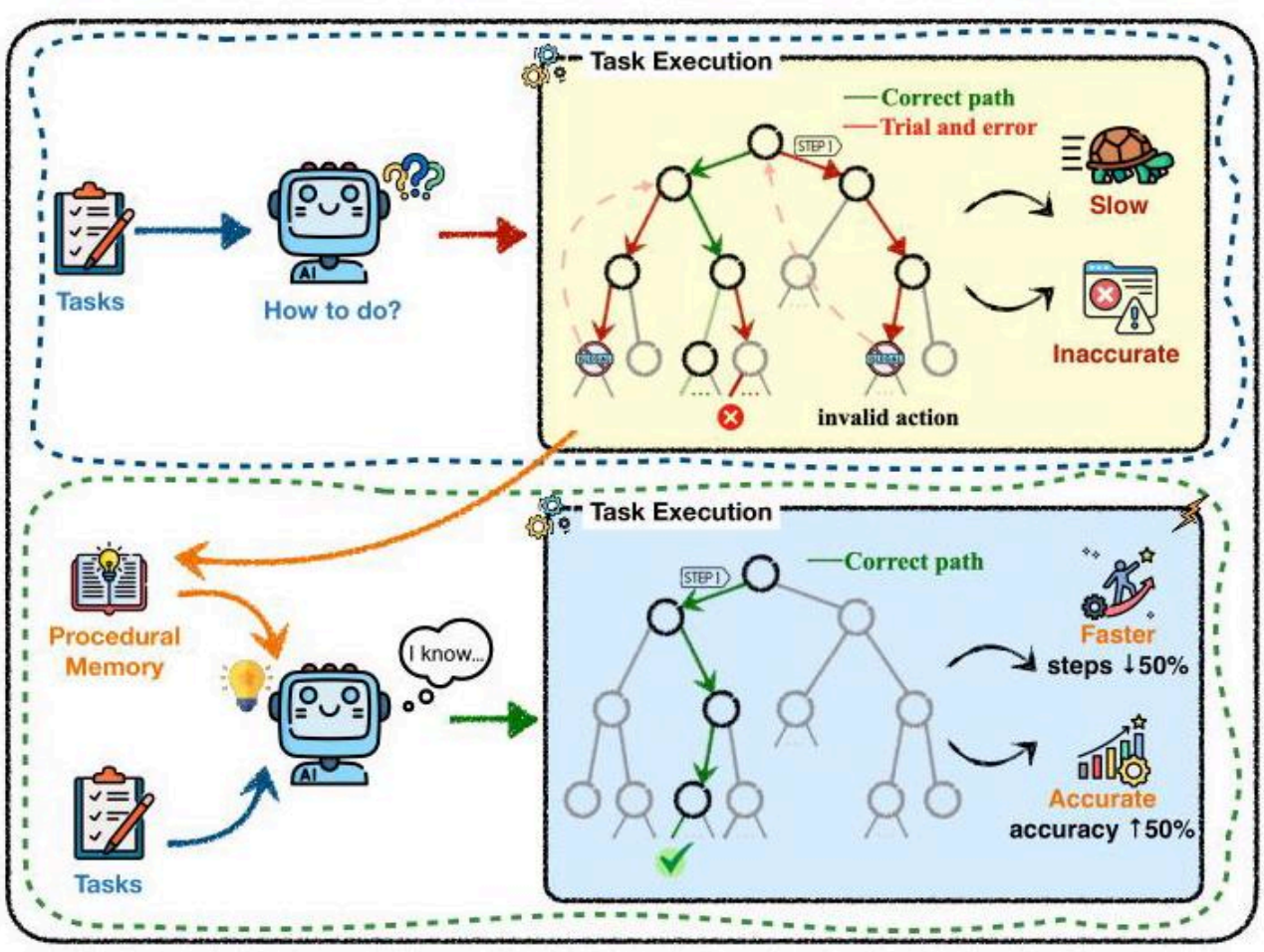

Figure 1: With procedural memory, agents can improve both the success rate (accuracy ↑) and execution efficiency (steps ↓) when solving similar tasks.
图1：借助程序性记忆，智能体在解决类似任务时，既能提升成功率（准确率 ↑），又能提高执行效率（步骤数 ↓ ）。

and a similar environment. Instead of starting fresh each time, an agent should extract its experience from past successes. By turning earlier trajectories into reusable templates like patterns of reasoning, tool sequences, and recovery tactics, it can progress step by step, learning from every failure and success, until even the most convoluted missions become routine.
和相似环境。智能体不应每次都从零开始，而应从过去的成功经验中提取教训。通过将早期轨迹转化为可复用的模板，如推理模式、工具序列和恢复策略，智能体可以逐步前进，从每次失败和成功中学习，直至最复杂的任务也变得常规化。

The capacity to distill, chronicle, and re-apply lessons from one's own experiential trajectory is the bedrock of human learning and the pivotal gateway through which an agent ascends toward self-directed refinement (Liu et al. 2025a; Sumers et al. 2023a; Li et al. 2023). Procedural memory (Gupta and Cohen 2002; Cohen and Squire 1980) silently compiles habitual skills into executable subroutines, enabling unconscious, fluent action. While contemporary agents built on LLMs can compose short action plans or call external tools, their procedural knowledge is either hand-crafted, stored as brittle prompt templates, or implicitly entangled in model parameters that are expensive to update. Existing memory-augmented frameworks such as LangGraph (Mavroudis 2024), AutoGPT (Yang, Yue, and He 2023), or agent cognitive architectures like
提炼、记录并重新应用自身经验轨迹中的教训的能力，是人类学习的基石，也是智能体迈向自主改进的关键通道（Liu等，2025a；Sumers等，2023a；Li等，2023）。程序性记忆（Gupta和Cohen，2002；Cohen和Squire，1980）默默地将习惯性技能编译成可执行的子程序，实现无意识且流畅的行动。尽管基于LLM的当代智能体能构建短期行动计划或调用外部工具，但其程序性知识要么是手工制作的、存储为脆弱的提示模板，要么隐含于难以更新的模型参数中。现有的记忆增强框架如LangGraph（Mavroudis，2024）、AutoGPT（Yang, Yue, 和 He，2023）或智能体认知架构如

- Equal Core Contributors.
- 共同核心贡献者。

† Corresponding Author.
† 通讯作者。

Memory Bank (Zhong et al. 2024a; Sumers et al. 2023b) and Soar (Laird 2022) provide coarse abstractions (buffers, rule chunks, production systems) but leave the optimization of procedural memory life-cycle operations about how skills are built, indexed, patched, and eventually pruned, largely unexamined. Consequently, there is no principled way to quantify how efficiently an agent evolves its procedural repertoire or to guarantee that new experiences improve rather than erode performance.
Memory Bank（Zhong 等，2024a；Sumers 等，2023b）和 Soar（Laird 2022）提供了粗略的抽象（缓冲区、规则块、产生系统），但对程序性记忆生命周期操作的优化——即技能如何构建、索引、修补及最终修剪——几乎未加探讨。因此，目前尚无系统方法来量化智能体如何高效演化其程序性技能库，也无法保证新经验是提升而非削弱性能。

To close this gap,we present Mem^p ,a task-agnostic framework that treats procedural memory as a first-class optimization object. The core exploration of Mem^p lies in how different strategies for memory construction, retrieval, and updating affect overall performance. During the construction phase, we follow the majority of traditional memory architectures and agent-based memory designs by leveraging either the full historical trajectory or explicit guidelines to guide the process. In the retrieval phase, we experiment with various key-building strategies-such as query-vector matching and keyword-vector matching-to investigate how procedural memory can be constructed more precisely. Unlike prior memory mechanisms or learning from experience, Mem^p introduces diverse procedural-memory update strategies: In the realm of agents, memory updating is crucial for agents to adapt to dynamic environments. By incorporating diverse strategies like ordinary addition, validation filtering, reflection, and dynamic discarding, agents can efficiently manage their knowledge base. This ensures they stay updated with new information, discard outdated data, and optimize memory resources. Such strategies enhance learning efficiency, improve decision-making quality, and boost adaptability, allowing agents to perform optimally in various tasks and scenarios.
为弥补这一空白，我们提出了Mem^p，一个任务无关的框架，将程序性记忆视为一等优化对象。Mem^p的核心探索在

于不同的记忆构建、检索和更新策略如何影响整体性能。在构建阶段，我们遵循大多数传统记忆架构和基于智能体的记忆设计，利用完整的历史轨迹或明确的指导原则来引导过程。在检索阶段，我们尝试多种关键构建策略——如查询向量匹配和关键词向量匹配——以探究如何更精确地构建程序性记忆。不同于以往的记忆机制或经验学习，Mem^p引入了多样的程序性记忆更新策略：在智能体领域，记忆更新对于适应动态环境至关重要。通过融合普通添加、验证过滤、反思和动态丢弃等多种策略，智能体能够高效管理其知识库，确保及时更新新信息，剔除过时数据，优化记忆资源。这些策略提升了学习效率、决策质量和适应能力，使智能体能在多样任务和场景中表现最佳。

We instantiate Mem^p on top of strong LLMs (GPT-4o and Claude, Qwen) and evaluate on two diverse domains: long-horizon housework ALFWorld (Shridhar et al. 2021) and long-term information seeking task TravelPlanner (Xie et al. 2024). On two benchmark datasets that rigorously evaluate agent capabilities, we demonstrate that constructing and retrieving procedural memory during training empowers an agent to distill and reuse its prior experience. When this memory is exploited at test time, the agent's task accuracy rises, and compared with tackling each instance in isolation, it eliminates most fruitless exploration on unfamiliar tasks, yielding substantial reductions in both step count and token consumption. Further, by equipping the agent with a set of memory-update mechanisms, we allow it to build and refine its procedural memory while acting in the test environment. This endows the agent with a continual, almost linear mastery of the task. Extensive ablations reveal that procedural memory also scales gracefully and transfers effectively to new, related tasks.
我们在强大的大型语言模型（GPT-4o 和 Claude、Qwen）基础上实现了Mem^p，并在两个不同领域进行评估：长时程家务任务 ALFWorld（Shridhar 等，2021）和长期信息检索任务 TravelPlanner（Xie 等，2024）。在两个严格评估智能体能力的基准数据集上，我们展示了在训练期间构建和检索程序性记忆使智能体能够提炼并复用先前经验。当测试时利用该记忆，智能体的任务准确率提升，并且相比于孤立处理每个实例，显著减少了在陌生任务上的无效探索，大幅降低了步骤数和令牌消耗。此外，通过为智能体配备一套记忆更新机制，我们使其能够在测试环境中构建和完善程序性记忆，从而赋予智能体持续、近乎线性的任务掌握能力。大量消融实验表明，程序性记忆具有良好的扩展性，并能有效迁移到新的相关任务。

5 Related Works

6 相关工作

Memory in Language Agents. Memory is a foundational component in language agents, enabling them to retain and utilize past information across multiple timescales, including short-term, episodic, and long-term memory, to enhance their performance and adaptability (Zhou et al. 2023,
语言智能体中的记忆。记忆是语言智能体的基础组成部分，使其能够跨多个时间尺度保留和利用过去信息，包括短期记忆、情节记忆和长期记忆，从而提升其性能和适应性（Zhou 等，2023，

2024; Zhang et al. 2024; Liu et al. 2025a; Li et al. 2025). These systems aim to mimic aspects of human memory to improve coherence, personalization, and learning capabilities (Chhikara et al. 2025; Wu et al. 2025b; Xia et al. 2025). Current approaches include end-to-end memory systems (Yu et al. 2025; Zhou et al. 2025), external memory systems (Chhikara et al. 2025; Zhong et al. 2024b), and hierarchical memory structures (Hu et al. 2024a; Xu et al. 2025). These methods involve encoding and storing information in various formats, using retrieval mechanisms like vector embeddings and semantic search, and implementing memory updating and forgetting strategies to maintain relevance and efficiency. Despite its importance, memory in multiturn agent interactions remains underexplored, and enabling agents to effectively learn and utilize memory across trajectories poses a significant challenge.
Procedural memory is a type of long-term memory that involves the retention of procedures and skills, such as typing or riding a bike, which are performed automatically without conscious thought. The agent utilizes procedural memory to internalize and automate repetitive tasks, decision-making processes, and interaction patterns, leading to more efficient and context-aware responses over time. Although there have been several works, such as Voyager (Wang et al. 2023), AWM (Wang et al. 2024b), and AutoManual (Chen et al. 2024), that utilize procedural memory to enhance agents' capabilities on similar tasks, there still lacks a systematic analysis on how to construct, retrieve, and update such procedural memory like (Wu et al. 2024). Therefore, our work mainly focuses on exploring how to

build an effective procedural memory system for agents performing cross-trajectory tasks.
2024；Zhang 等，2024；Liu 等，2025a；Li 等，2025）。这些系统旨在模拟人类记忆的某些方面，以提升连贯性、个性化和学习能力（Chhikara 等，2025；Wu 等，2025b；Xia 等，2025）。当前的方法包括端到端记忆系统（Yu 等，2025；Zhou 等，2025）、外部记忆系统（Chhikara 等，2025；Zhong 等，2024b）和分层记忆结构（Hu 等，2024a；Xu 等，2025）。这些方法涉及以多种格式编码和存储信息，利用向量嵌入和语义搜索等检索机制，并实施记忆更新与遗忘策略以保持相关性和效率。尽管记忆在多轮代理交互中至关重要，但仍未得到充分研究，使代理能够有效学习并利用跨轨迹记忆仍是一大挑战。程序性记忆是一种长期记忆，涉及程序和技能的保留，如打字或骑自行车，这些行为无需有意识思考即可自动执行。代理利用程序性记忆内化并自动化重复任务、决策过程和交互模式，从而随着时间推移实现更高效且具上下文感知的响应。尽管已有诸多工作，如 Voyager（Wang 等，2023）、AWM（Wang 等，2024b）和 AutoManual（Chen 等，2024），利用程序性记忆提升代理在类似任务上的能力，但仍缺乏关于如何构建、检索和更新此类程序性记忆的系统性分析（Wu 等，2024）。因此，我们的工作主要聚焦于探索如何为执行跨轨迹任务的代理构建有效的程序性记忆系统。

Learning from Experience. LLM-based Agent learning from experience involves intelligence continuously improving their decision-making capabilities through interaction with environments and utilization of past experiences (Tan et al. 2025; Tang et al. 2025; Zhou et al. 2025; Qiao et al. 2025; Su et al. 2025; Wang et al. 2024b). This approach is crucial for developing adaptive and intelligent agents capable of handling dynamic real-world scenarios, as it allows them to optimize behaviors, reduce manual programming needs, and enhance performance across various tasks (Zheng et al.; Liu et al. 2025c; Wang et al. 2025). Agents typically employ mechanisms such as reinforcement learning (Lu et al. 2025; Dong et al. 2025), experience replay (Feng et al. 2025; Liu et al. 2025b), imitation learning (Sun et al. 2024; Yang et al. 2024b), memory management (Hou, Tamoto, and Miyashita 2024; Hu et al. 2024b), and multi-agent learning to achieve this. However, current methods face limitations including low sample efficiency, poor generalization across tasks, catastrophic forgetting when learning new information, and there are very few features for memory update. The key distinction of our work lies in systematically investigating optimal strategies for construction, retrieval, and update modules of an agent's procedural knowledge. During the update phase, we enhance the agent's capabilities by maintaining an editable repository of procedural knowledge. Additionally, collecting high-quality training data can be challenging and may introduce biases. Addressing these limi-
从经验中学习。基于大语言模型（LLM）的代理从经验中学习，指智能体通过与环境交互并利用过去经验持续提升其决策能力（Tan 等，2025；Tang 等，2025；Zhou 等，2025；Qiao 等，2025；Su 等，2025；Wang 等，2024b）。该方法对于开发能够应对动态现实场景的自适应智能代理至关重要，因为它使代理能够优化行为、减少人工编程需求并提升多任务性能（Zheng 等；Liu 等，2025c；Wang 等，2025）。代理通常采用强化学习（Lu 等，2025；Dong 等，2025）、经验回放（Feng 等，2025；Liu 等，2025b）、模仿学习（Sun 等，2024；Yang 等，2024b）、记忆管理（Hou、Tamoto 和 Miyashita，2024；Hu 等，2024b）以及多代理学习等机制来实现这一目标。然而，当前方法存在样本效率低、跨任务泛化能力差、学习新信息时灾难性遗忘以及记忆更新功能匮乏等限制。我们的工作关键区别在于系统性地研究代理程序性知识构建、检索和更新模块的最优策略。在更新阶段，我们通过维护可编辑的程序性知识库来增强代理能力。此外，高质量训练数据的收集具有挑战性，且可能引入偏差。解决这些限-

tations is essential for advancing the capabilities of LLM-based agents and ensuring their effective application in real-world contexts.
制对于提升基于LLM的代理能力并确保其在现实环境中的有效应用至关重要。

7 Preliminary

8 初步

When an agent influences its external environment by invoking external tools or executing prescribed actions, and iteratively refines its behavior over multiple rounds to accomplish a complex multi-step objective, this paradigm can be modeled as a Markov Decision Process (MDP). (Puter-man 1990) Under this view,at each discrete time step t ,the agent,situated in state $s_t \in S$,chooses an action $a_t \in A$, according to its policy $\pi(a_t \mid s_t)$,where A is the action space of the task. The environment then transitions to a new state $s_{t+1} \in S$ and emits an observation O_t .

Consequently,the entire interaction trajectory may be compactly expressed as:
当代理通过调用外部工具或执行预定动作影响其外部环境，并通过多轮迭代不断优化行为以完成复杂的多步骤目标时，该范式可建模为马尔可夫决策过程（MDP）（Puterman，1990）。在此视角下，每个离散时间步 t ，代理处于状态 $s_t \in S$ ，根据其策略 $\pi(a_t \mid s_t)$ 选择动作 $a_t \in A$ ，其中 A 是任务的动作空间。环境随后转移到新状态 $s_{t+1} \in S$ 并发出观测 O_t 。因此，整个交互轨迹可简洁表示为：

$$\tau = (s_0, a_0, o_1, s_1, a_1, o_2, \ldots, s_T), \tag{1}$$

where τ is the complete exploration trajectory of this task. Moreover,a reward function R will evaluate the task's completion r within this environment env by assigning a score based on the final state s_T or the entire trajectory τ .
其中 τ 是该任务的完整探索轨迹。此外，奖励函数 R 将通过基于最终状态 s_T 或整个轨迹 τ 赋分，评估该环境 env 中任务的完成度 r 。

$$r = R(env, s_T, \tau) \in [0, 1] \tag{2}$$

Although approaches resembling Markov Decision Processes inevitably contain erroneous actions and exploratory attempts, the contextual information they generate becomes valuable for decision-making as the model's reasoning and reflective capabilities improve. Nevertheless, this benefit comes at a high test-time cost-both in time and in token consumption. When facing an entirely new and complex environment, many actions (or tokens) are spent simply understanding the environment and the task itself. This leads to redundancy when similar tasks are executed within the same environment: the agent has already acquired partial procedural knowledge about the environment or task during earlier episodes, yet fails to transfer that knowledge effectively to subsequent tasks.
尽管类似马尔可夫决策过程（Markov Decision Processes）的方法不可避免地包含错误动作和探索尝试，但随着模型推理和反思能力的提升，它们生成的上下文信息在决策中变得有价值。然而，这种收益伴随着高昂的测试时成本——无论是时间还是令牌消耗。在面对全新且复杂的环境时，许多动作（或令牌）仅用于理解环境和任务本身。这导致在同一环境中执行类似任务时出现冗余：代理在早期回合中已部分掌握了环境或任务的程序性知识，但未能有效地将该知识迁移到后续任务中。

By shifting from parallel task completion to sequential task completion, the agent can learn and distill experience from earlier tasks, thereby reducing repetitive exploration. Inspired by human procedural memory, we propose to equip the agent with a procedural memory module. This module transforms the conventional policy $\pi(a_t \mid s_t)$ into $\pi_{m^p}(a_t \mid s_t)$, where m^p is the agent's learned procedural memory.
通过从并行任务完成转向顺序任务完成，代理可以从早期任务中学习并提炼经验，从而减少重复探索。受人类程序性记忆启发，我们提出为代理配备一个程序性记忆模块。该模块将传统策略$\pi(a_t \mid s_t)$转化为$\pi_{m^p}(a_t \mid s_t)$，其中m^p是代理学习到的程序性记忆。

9 Agent Procedural Memory

10 代理程序性记忆

Procedural memory is the type of long-term memory responsible for knowing how to perform tasks and skills, such as typing or riding a bike. By mastering this type of procedural memory, humans avoid the need to relearn the process each time. For an agent,that is,for a task trajectory τ and its reward r ,a memory m^p is constructed by a builder B ,thereby achieving the acquisition of memory, namely
程序性记忆是一种负责知道如何执行任务和技能的长期记忆类型，例如打字或骑自行车。通过掌握这种程序性记忆，人类避免了每次都重新学习过程。对于代理而言，即对于任务轨迹τ及其奖励r，由构建器B构造记忆m^p，从而实现记忆的获取，即

$$\text{Mem} = \sum_{t=1}^{T} m^{p_t}, \text{ where } m^{p_t} = B(\tau_t, r_t) \tag{3}$$

where Mem is the procedural memory library acquired by the agent over the T tasks. After constructing the procedural memory library,when facing a new task t_{new} ,we need a good procedural memory retriever to recall a memory that fits t_{new} . Generally speaking,we would choose the task $t \in T$ that is most similar to t_{new} ,because similar experiences are more helpful for the agent to complete the new task.
其中Mem是代理在T任务中获得的程序性记忆库。构建程序性记忆库后，面对新任务t_{new}时，我们需要一个良好的程序性记忆检索器来回忆与t_{new}匹配的记忆。一般来说，我们会选择与t_{new}最相似的任务$t \in T$，因为相似的经验对代理完成新任务更有帮助。

$$m_{\text{retrieved}} = \arg \max_{m^{Pi} \in Mem} S\left(t_{\text{new}}, t_i\right) \tag{4}$$

As we use cosine similarity for the vector embedding model ϕ of the task in the experiment,the retrieval process becomes:
在实验中，我们对任务的向量嵌入模型ϕ使用余弦相似度，检索过程变为：

$$m_{\text{retrieved}} = \arg \max_{m^{Pi} \in Mem} \frac{\phi\left(t_{\text{new}}\right) \cdot \phi\left(t_i\right)}{\| \phi\left(t_{\text{new}}\right) \| \|\phi\left(t_i\right)\|}. \tag{5}$$

Moreover, as the number of completed tasks continuously increases, simply augmenting the agent's procedural memory is inconsistent with common sense. A well-designed procedural memory system should have a reasonable update mechanism-that is, it should dynamically perform addition, deletion, modification, and retrieval based on the task execution context.
此外，随着完成任务数量的不断增加，单纯增加代理的程序性记忆与常识不符。一个设计良好的程序性记忆系统应具备合理的更新机制——即应根据任务执行上下文动态执行添加、删除、修改和检索操作。

Let $M(t)$ denote the agent's procedural memory at time t ,and τ_t represent the set of tasks completed up to time t . Then,the update mechanism can be modeled as a function U that takes the current procedural memory and task execution feedback to produce the updated memory:
设$M(t)$表示时间点t代理的程序性记忆，τ_t表示截至时间点t完成的任务集合。则更新机制可建模为函数U，其以当前程序性记忆和任务执行反馈为输入，生成更新后的记忆：

$$M(t+1) = U(M(t), E(t), \tau_t), \tag{6}$$

where $E(t)$ encapsulates the execution feedback (e.g.,success, failure, performance metrics). A more sophisticated implementation of U could be represented as:
其中$E(t)$封装执行反馈（如成功、失败、性能指标）。U的更复杂实现可表示为：

$$U = \text{Add}\left(M_{\text{new}}\right) \ominus \text{Remove}\left(M_{\text{obso}}\right) \oplus \text{Update}\left(M_{\text{exist}}\right),$$

(7)

where M_{new} represents new procedural memory to be added; M_{obso} indicates procedural memory to be removed, M_{existing} are tasks to be updated based on execution feedback $E(t)$. This comprehensive formula captures the essential add, delete, and modify operations within the update mechanism.
其中M_{new}表示待添加的新程序性记忆；M_{obso}表示待删除的程序性记忆，M_{existing}为基于执行反馈$E(t)$需更新的任务。该综合公式涵盖了更新机制中的添加、删除和修改操作的核心内容。

11 Experiment

12 实验

In this section, we will introduce the Procedural Memory framework in detail (Figure 2), covering the storage, retrieval, and update modules of memory, as well as analyzing which strategies perform better within each module.
本节将详细介绍程序性记忆框架（图2），涵盖记忆的存储、检索和更新模块，并分析各模块中表现更优的策略。

13 Experimental Settings

14 实验设置

Datasets. For our experiments, we adapt TravelPlan-ner (Xie et al. 2024) and ALFWorld (Shridhar et al. 2021) benchmarks. TravelPlanner is a benchmark designed to evaluate agents' ability to use tools and perform complex planning under intricate constraints. In contrast, ALFWorld comprises household tasks. In each interaction round, the agent outputs an action, and the environment responds with textual feedback describing the resulting state. This process repeats for multiple turns until the task is completed or the maximum number of rounds is reached. ALFWorld includes test split to evaluate the agent's generalization ability.
数据集。我们的实验采用TravelPlan-ner（谢等，2024）和ALFWorld（Shridhar等，2021）基准。TravelPlanner是一个评估代理使用工具和在复杂约束下进行复杂规划能力的基准。相比之下，ALFWorld包含家务任务。在每轮交互中，代理输出一个动作，环境以文本反馈描述结果状态。该过程重复多轮，直至任务完成或达到最大轮数。ALFWorld包含测试集以评估代理的泛化能力。

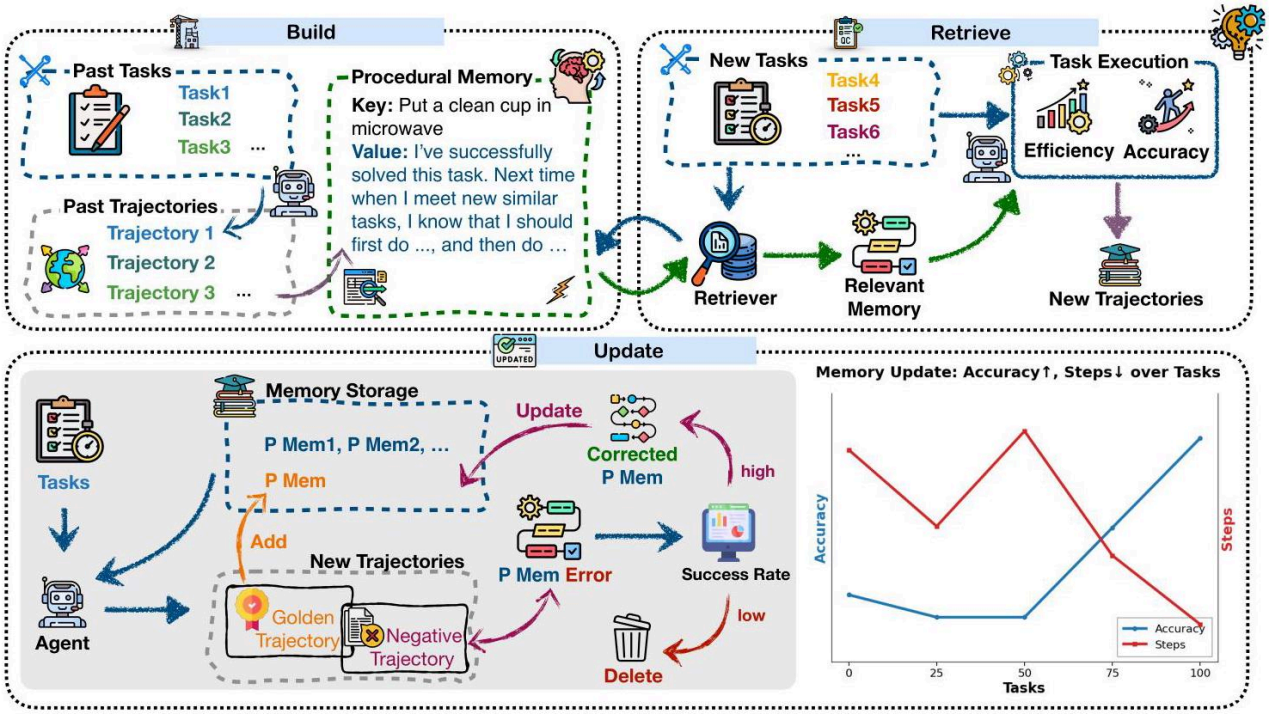

Figure 2: The procedural memory framework consists of Build, Retrieve, and Update, which respectively involve encoding stored procedural memory, forming new procedural memories, and modifying existing ones in light of new experiences.
图2：程序性记忆框架包括构建、检索和更新，分别涉及编码存储的程序性记忆、形成新的程序性记忆以及根据新经验修改现有记忆。

Backbones. In our experiments, we benchmarked our procedural memory on three base models. Specifically, we adopt the two proprietary frontier models that have consistently dominated public leaderboards: OpenAI's GPT- 4o (OpenAI 2022) and Anthropic's Claude (Anthropic 2022), and complement them with the open-sourced Qwen2.5-72B-Instruct (Yang et al. 2024a). The first two provide state-of-the-art closed-source performance, while the third allows us to verify that our findings generalize beyond proprietary systems and remain valid in the open-source regime.
骨干网络。在我们的实验中，我们在三种基础模型上对程序性记忆进行了基准测试。具体来说，我们采用了两种始终主导公共排行榜的专有前沿模型：OpenAI的GPT-4o（OpenAI 2022）和Anthropic的Claude（Anthropic 2022），并辅以开源的Qwen2.5-72B-Instruct（Yang等，2024a）。前两者提供了最先进的闭源性能，而第三者则使我们能够验证我们的发现是否超越专有系统，在开源环境中依然有效。

Evaluation. For ALFWorld dataset, task completion is evaluated by the execution environment. After a task is completed or the maximum number of execution steps is reached, the environment provides a reward of 0 or 1 to indicate whether the task has been successfully completed. For TravelPlanner, we conduct experiments on the test set in a two-stage mode. After multiple rounds of interaction to obtain the travel trajectory and the final planner, GPT-40 converts the travel plan into a specified JSON format. The converted plan is then compared with the gold standard to obtain scores for both Common Sense and Hard Constraint.
评估。对于ALFWorld数据集，任务完成情况由执行环境评估。任务完成或达到最大执行步数后，环境会提供0或1的奖励，以指示任务是否成功完成。对于TravelPlanner，我们在测试集上以两阶段模式进行实验。经过多轮交互获取旅行轨迹和最终规划后，GPT-40将旅行计划转换为指定的JSON格式。转换后的计划随后与标准答案进行比较，以获得常识和硬约束两项评分。

15 Memory Storage & Retrieval

16 记忆存储与检索

Procedural knowledge is typically stored in two main formats: (1) trajectories are kept verbatim, round by round, in memory, or (2) high-level abstractions are extracted from these trajectories and then stored. Once a similar procedu-
程序性知识通常以两种主要格式存储：（1）逐轮逐字保留轨迹，存于记忆中，或（2）从这些轨迹中提取高级抽象后存储。一旦检索到相似的程序性记忆，

ral memory is retrieved, it is appended to the task as part of the context, serving as prior knowledge to assist the model in completing the task.
它会作为上下文的一部分附加到任务中，作为先验知识辅助模型完成任务。

Inspired by this, we designed the following experimental conditions:
受此启发，我们设计了以下实验条件：

- No Memory: The model tackles the assigned task in a ReAct fashion without any external memory.
- 无记忆：模型以ReAct方式处理分配的任务，不使用任何外部记忆。

- Trajectory: We first filter the gold trajectories from the training set and store them. At inference time, the system retrieves the top-k trajectories whose query vectors are most similar to the current task's vector, supplying them as procedural memories before execution.
- 轨迹：我们首先从训练集中筛选出标准轨迹并存储。在推理时，系统检索与当前任务向量最相似的前k条轨迹，作为程序性记忆提供给执行过程。

- Script: The model analyzes and summarizes the gold trajectories from the training set, distilling them into abstract procedural knowledge that is provided as a prompt before each task.
- 脚本：模型分析并总结训练集中的标准轨迹，将其提炼为抽象的程序性知识，作为每个任务前的提示提供。

- Proceduralization: This condition combines the full retrieved trajectories with the high-level script generated by the model, integrating both concrete examples and abstract guidance as the procedural memory.
- 程序化：该条件结合了完整检索的轨迹和模型生成的高级脚本，整合具体示例与抽象指导作为程序性记忆。

As shown in Table 1, all memory construction methods outperform the no-memory baseline, achieving higher scores on both datasets while also reducing the number of steps required. This indicates that procedural memory built during training is beneficial for directly applying tasks during testing. Furthermore, we observe that the approach of abstracting trajectories into scripts during training yields rel-
如表1所示，所有记忆构建方法均优于无记忆基线，在两个数据集上均取得更高分数，同时减少所需步骤数。这表明训练期间构建的程序性记忆有助于测试时直接应用任务。此外，我们观察到训练期间将轨迹抽象为脚本的方法带来了相对优势——

Model	Granularity	TravelPlanner			ALFWorld		
		#CS↑	#HC↑	Steps $\downarrow$	Dev $\uparrow$	Test↑	Steps $\downarrow$
GPT-40	No Memory	71.93	12.88	17.84	39.28	42.14	23.76
	Script	72.08	5.50	15.79	66.67	56.43	18.52
	Trajectory	76.02	8.25	14.64	67.17	74.29	16.49
	Proceduralization	79.94	9.76	14.62	87.14	77.86	15.01
Claude-3.5-sonnet	No Memory	63.49	33.06	18.84	39.20	34.97	24.12
	Script	62.08	29.61	19.21	56.13	53.59	19.38
	Trajectory	65.76	29.61	17.72	69.28	71.78	15.97
	Proceduralization	65.46	30.14	15.29	82.50	74.72	15.79
Qwen2.5-72b	No Memory	56.57	7.34	18.32	44.91	41.25	21.38
	Script	58.59	7.34	18.53	66.24	61.88	17.13
	Trajectory	63.41	12.66	18.12	64.49	69.57	16.40
	Proceduralization	63.82	14.19	17.94	85.71	77.19	15.32

模型	粒度	旅行规划器			ALFWorld		
		#CS↑	#HC↑	步骤 $\downarrow$	开发 $\uparrow$	测试↑	步骤 $\downarrow$
GPT-40	无记忆	71.93	12.88	17.84	39.28	42.14	23.76
	脚本	72.08	5.50	15.79	66.67	56.43	18.52
	轨迹	76.02	8.25	14.64	67.17	74.29	16.49
	程序化	79.94	9.76	14.62	87.14	77.86	15.01
Claude-3.5-sonnet	无记忆	63.49	33.06	18.84	39.20	34.97	24.12
	脚本	62.08	29.61	19.21	56.13	53.59	19.38
	轨迹	65.76	29.61	17.72	69.28	71.78	15.97
	程序化	65.46	30.14	15.29	82.50	74.72	15.79
Qwen2.5-72b	无记忆	56.57	7.34	18.32	44.91	41.25	21.38
	脚本	58.59	7.34	18.53	66.24	61.88	17.13
	轨迹	63.41	12.66	18.12	64.49	69.57	16.40
	程序化	63.82	14.19	17.94	85.71	77.19	15.32

Table 1: Results on Build Policy. #CS, #HC denote Commensense and Hard Constraint, respectively. ↑ indicates the higher values are better, and ↓ denotes the lower values are better. The best results among all methods with similar settings are bolded, and the second-best results are underlined.
表1：构建策略的结果。#CS、#HC分别表示常识（Commensense）和硬约束（Hard Constraint）。↑表示数值越大越好，↓表示数值越小越好。所有方法中在相似设置下的最佳结果加粗，次佳结果加下划线。

atively better performance on the ALFWorld test set compared to the dev set. Conversely, trajectories that utilize complete execution traces as procedural memory achieve higher scores on the dev set, suggesting that scripts are more capable of generalizing to different test tasks, while trajectories are better suited for scenarios involving tasks similar to those already completed. By combining procedure knowledge from both methods of employing abstracted guidelines along with concrete execution trajectories, we attain the optimal performance.
在ALFWorld测试集上表现相较于开发集有较好的提升。相反，利用完整执行轨迹作为程序记忆的轨迹在开发集上得分更高，表明脚本（scripts）更能泛化到不同的测试任务，而轨迹更适合处理与已完成任务相似的场景。通过结合抽象指导和具体执行轨迹两种方法的程序知识，我们达到了最佳性能。

After converting a set of completed trajectories into procedural memory, the next critical challenge is to retrieve the most accurate and relevant procedural knowledge when a new task arrives. We have designed several different key construction methods for memory storage to facilitate subsequent vector-based matching and retrieval:
将一组完成的轨迹转换为程序记忆后，下一个关键挑战是在新任务到来时检索最准确且相关的程序知识。我们设计了几种不同的键构造方法用于记忆存储，以便后续基于向量的匹配和检索：

- Random Sample: Does not utilize keys for vector retrieval; instead, randomly extracts a few memories from procedural memory.
- 随机采样：不使用键进行向量检索，而是从程序记忆中随机抽取若干记忆。

- Query: Employ query description as the key for storage, leveraging the semantic similarity of queries for retrieval.
- 查询：使用查询描述作为存储键，利用查询的语义相似性进行检索。

- AveFact: We apply a large model to extract keywords from the task's query, then computes the average similarity across matched keywords for retrieval.
- AveFact：我们应用大型模型从任务查询中提取关键词，然后计算匹配关键词的平均相似度进行检索。

During the retrieval process, we evaluate the similarity by calculating the cosine similarity between their corresponding vectors. Our experiments show that these different retrieval strategies produce varying results. Specifically, compared to random sampling, employing the query based and AveFact methods for precise retrieval significantly improves performance. The query-based approach benefits from capturing semantic contexts, enabling more accurate matches. The AveFact method, by extracting key features and averaging their similarities, effectively focuses on core task elements, leading to better retrieval efficacy. Overall, our findings suggest that incorporating semantic understanding and key feature extraction in retrieval strategies substantially enhances memory access accuracy and the effectiveness of downstream task performance.
在检索过程中，我们通过计算对应向量的余弦相似度来评估相似性。实验表明，不同的检索策略产生了不同的结果。具体来说，与随机采样相比，采用基于查询和AveFact方法的精确检索显著提升了性能。基于查询的方法通过捕捉语义上下文，实现了更准确的匹配。AveFact方法通过提取关键特征并平均其相似度，有效聚焦于任务核心元素，提升了检索效果。总体来看，我们的研究表明，在检索策略中融入语义理解和关键特征提取，显著增强了记忆访问的准确性和下游任务的执行效果。

Model	Policy	#CS↑ $\# {HC} \uparrow$ Steps $\downarrow$		
GPT-40	No Memory	71.93	12.88	17.84
	Random Sample	74.59	6.72	15.12
	Key=Query	73.38	8.95	15.44
	Key=AveFact	76.02	8.25	14.64
Claude-3.5-sonnet	No Memory	63.49	33.06	18.84
	Random Sample	63.99	29.91	17.93
	Key=Query	64.93	28.56	17.60
	Key=AveFact	65.76	29.61	17.72
Qwen2.5-72b	No Memory	56.57	7.34	18.32
	Random Sample	59.76	8.43	18.31
	Key=Query	61.71	11.97	18.54
	Key=AveFact	63.41	12.66	18.12
模型	策略	#CS↑ $\# {HC} \uparrow$ 步骤 $\downarrow$		
GPT-40	无记忆	71.93	12.88	17.84
	随机采样	74.59	6.72	15.12
	键=查询	73.38	8.95	15.44
	键=平均因子	76.02	8.25	14.64
Claude-3.5-十四行诗	无记忆	63.49	33.06	18.84
	随机采样	63.99	29.91	17.93
	键=查询	64.93	28.56	17.60
	键=平均因子	65.76	29.61	17.72
Qwen2.5-72b	无记忆	56.57	7.34	18.32
	随机采样	59.76	8.43	18.31
	键=查询	61.71	11.97	18.54
	键=平均因子	63.41	12.66	18.12

Table 2: Results on Retrieve Policy on TravelPlanner.
表2：TravelPlanner上检索策略的结果。

17 Memory Update

18 记忆更新

While many prior efforts have focused on developing reusable procedural knowledge, enabling models to learn from prior experiences rather than solving each test task in isolation, most existing memory update methods remain quite rudimentary. Typically, they simply append newly acquired memories to the existing store-a so-called "merge" strategy. In this work, we explore several online memory-update mechanisms to identify which dynamic strategy delivers the best performance on our tasks. Beyond end-to-end evaluation metrics, we also analyze how both accuracy and efficiency evolve as the number of executed tasks increases, explicitly measuring the benefits conferred by our procedural memory.
尽管许多先前的研究致力于开发可复用的程序性知识，使模型能够从以往经验中学习，而非孤立地解决每个测试任务，但现有的大多数记忆更新方法仍相当初级。通常，它们只是将新获得的记忆简单地附加到现有存储中——所谓的“合并”策略。在本研究中，我们探索了几种在线记忆更新机制，以确定哪种动态策略在我们的任务中表现最佳。除了端到端的评估指标外，我们还分析了随着执行任务数量增加，准确率和效率的变化，明确衡量了程序性记忆带来的益处。

To facilitate systematic comparison, we designed several memory-update scenarios. In each, the agent's episodic memory is refreshed after every t test-set tasks. The specific update strategies are as follows:
为了便于系统比较，我们设计了若干记忆更新场景。在每个场景中，代理的情景记忆在每执行完t个测试集任务后刷新。具体的更新策略如下：

- Vanilla Memory Update: After every t tasks,all trajectories from these tasks are consolidated into procedural memories and directly appended to the memory bank.
- 基础记忆更新：每完成t个任务后，将这些任务的所有轨迹整合为程序性记忆，直接附加到记忆库中。

- Validation: After every t tasks,only the trajectories of
- 验证：每完成t个任务后，仅保留成功完成任务的轨迹，并将其转换为程序性记忆进行存储。

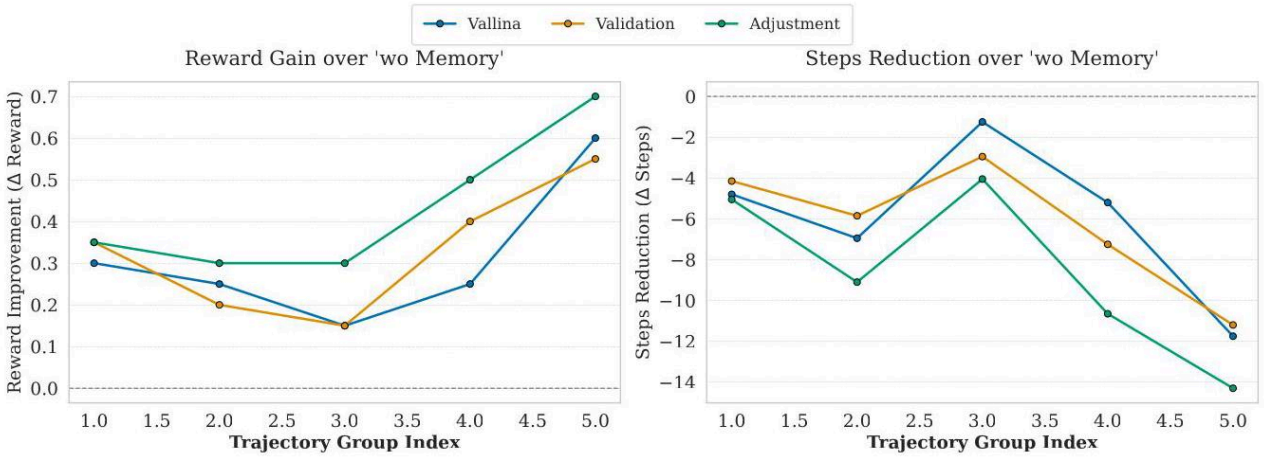

Figure 3: Reward gain and steps reduction vs. trajectory group index with procedural memory.
图3：使用程序性记忆时，奖励增益和步骤减少与轨迹组索引的关系。

successfully completed tasks are retained and converted into procedural memories for storage.
成功完成的任务轨迹被保留并转换为程序性记忆进行存储。

- Adjustment: When a retrieved procedural memory results in a failed execution, the erroneous trajectory is combined with the original memory and then revised in place, yielding an updated procedural memory.
- 调整：当检索到的程序性记忆导致执行失败时，将错误轨迹与原始记忆结合并在原地修正，生成更新后的程序性记忆。

As depicted in Figure 3, we systematically divided the tasks within our testbed into several distinct groups, with each group comprising a diverse set of individual tasks. Upon the completion of tasks within each group, we employed the previously described strategies to construct, store, and update the procedural memory. The experimental results reveal a clear trend: as we sequentially progress through more groups and iteratively refresh the memory, all strategies contribute to improved performance on subsequent tasks. Specifically, this is reflected not only in higher overall scores but also in a reduction in the number of steps required to complete the tasks.
如图3所示，我们系统地将测试集中的任务划分为若干不同组，每组包含多样化的单个任务。完成每组任务后，我们采用上述策略构建、存储并更新程序性记忆。实验结果显示出明显趋势：随着我们依次完成更多组任务并迭代刷新记忆，所有策略均促进了后续任务性能的提升。具体表现为整体得分提高以及完成任务所需步骤减少。

A closer comparison of different strategies exposes significant disparities in their effectiveness. Notably, the reflexion-based update mechanism stands out as the most effective approach. By the time the final group of tasks is reached, this method delivers a substantial advantage: it surpasses the second-best strategy by an impressive margin of +0.7 points and achieves a reduction of 14 steps. These improvements underscore the value of continually updating the memory, particularly when the update is guided by an error-correction mechanism embedded in the

reflexion process.
对不同策略的深入比较揭示了其效果的显著差异。值得注意的是，基于反思（reflexion）的更新机制表现最为出色。到达最后一组任务时，该方法带来了显著优势：其得分比第二佳策略高出0.7分，且步骤减少了14步。这些改进凸显了持续更新记忆的价值，尤其是在更新过程中嵌入了错误修正机制的反思过程指导下。

19 Analysis

20 分析

Procedural Memory Boosts Accuracy and Cuts Trials. Figure 5 presents a case study demonstrating how Procedural Memory enhances both accuracy and efficiency. In the absence of Procedural Memory, facing a complex task that has not been performed before, there are usually two situations. In the first scenario (left), the model repeatedly attempts illegal or incorrect actions, causing the context to become increasingly complex and eventually exceeding the model's understanding capacity. In the second scenario
程序性记忆提升准确率并减少尝试次数。图5展示了一个案例研究，说明程序性记忆如何增强准确性和效率。在缺乏程序性记忆的情况下，面对一个之前未执行过的复杂任务，通常有两种情况。第一种情形（左图），模型反复尝试非法或错误操作，导致上下文变得越来越复杂，最终超出模型的理解能力。第二种情形

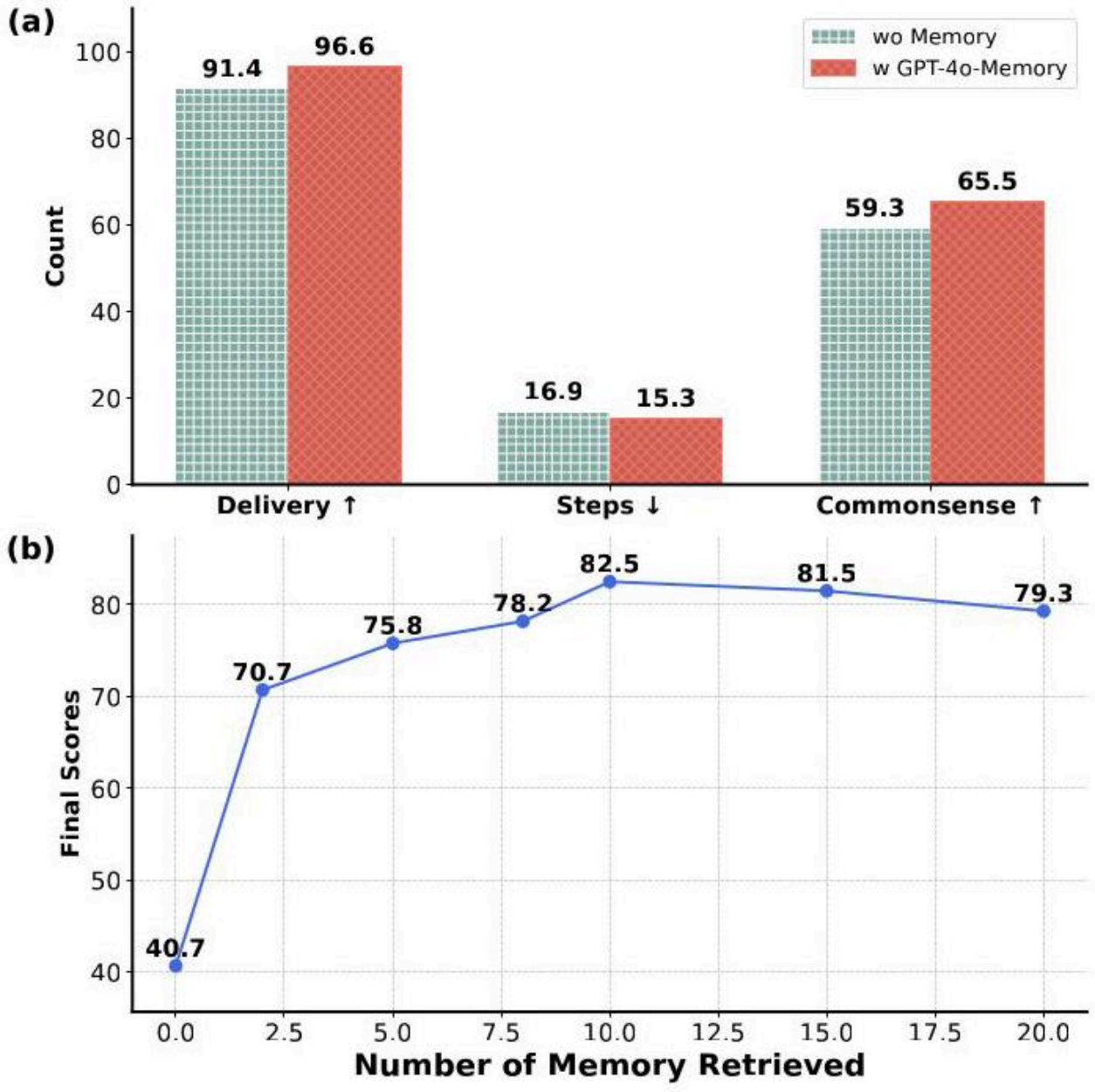

Figure 4: (a) Transfer result of GPT-4o's procedural memory to Qwen2.5-14B-Instruct and its performance on Trav-elPlanner dataset.(b) The relationship between the quantity of procedural memory retrieved for GPT-4o's performance on the ALFWorld dataset.
图4：（a）GPT-4o的程序性记忆迁移到Qwen2.5-14B-Instruct的结果及其在TravelPlanner数据集上的表现。（b）检索程序性记忆数量与GPT-4o在ALFWorld数据集上表现的关系。

(middle), after multiple attempts, the model completes the task but at a cost significantly higher than the optimal path.
（中图）经过多次尝试后，模型完成了任务，但所耗成本远高于最优路径。

In contrast, once Procedural Memory is available for similar tasks, the model spends less time on trial and error. For example, in the egg heating problem, Procedural Memory can indicate the approximate location of the egg, saving the aimless search. During the heating process, it provides clear guidance, ensuring that the heating actions are performed consecutively and correctly, thereby allowing the task to be completed in fewer steps.
相比之下，一旦针对类似任务具备程序性记忆，模型在试错上花费的时间就会减少。例如，在加热鸡蛋的问题中，程序性记忆可以指示鸡蛋的大致位置，避免盲目搜索。在加热过程中，它提供明确指导，确保加热动作连续且正确执行，从而使任务能在更少步骤内完成。

Figure 5: Compare trajectories with and without procedural memory, shortens the process by 9 steps and saves 685 tokens.
图5：比较有无程序记忆的轨迹，缩短了9步过程，节省了685个标记。

Procedural memory exhibits transferability from strong models to weaker ones. For a procedural memory constructed from a strong model in an offline memory library, we aim to verify whether this form of procedural memory can be effectively transferred to other models, or even weaker models. This exploration underscores the significance of memory transfer, as it could potentially enhance the adaptability and efficiency of various models by leveraging the knowledge and experience encapsulated within the strong model's memory structure. As shown in Figure 4 (b), procedural memory generated by GPT-4o was employed by Qwen2.5-14B. On the Travel Plan benchmark, the 14 billion parameter model raised its task completion rate by 5% and cut the average number of steps by 1.6. Similar gains, both in success rate and trajectory length, appeared on ALF-World. These outcomes confirm that procedural knowledge from a stronger model can be distilled into a reusable memory bank and

transferred to a smaller model with minimal overhead, giving that smaller model a clear boost in task solving ability. Moreover, by leveraging procedural memory transfer, we can rapidly migrate the experiential knowledge that one model has acquired to another, which is highly beneficial for agents as they adapt to new tasks with greater efficiency and robustness.
程序记忆表现出从强模型向弱模型的迁移能力。对于离线记忆库中由强模型构建的程序记忆，我们旨在验证这种程序记忆形式是否能有效迁移到其他模型，甚至更弱的模型。这一探索强调了记忆迁移的重要性，因为它有可能通过利用强模型记忆结构中封装的知识和经验，提升各种模型的适应性和效率。如图4(b)所示，GPT-4o生成的程序记忆被Qwen2.5-14B采用。在旅行计划基准测试中，该140亿参数模型的任务完成率提升了5%，平均步骤数减少了1.6。在ALF-World上也出现了成功率和轨迹长度的类似提升。这些结果证实，来自更强模型的程序知识可以被提炼成可复用的记忆库，并以极低的开销转移到较小模型，显著增强该模型的任务解决能力。此外，通过利用程序记忆迁移，我们可以快速将一个模型获得的经验知识迁移给另一个模型，这对于代理适应新任务，提高效率和鲁棒性极为有利。

Scaling Memory Retrieval Improves Agent Performance. While our main experiment has already demonstrated that procedural memory improves an agent's task accuracy and reduces the number of steps required, vector-based storage and retrieval confer an advantage over human procedural memory: they can be scaled both in total capacity and in the number of memories retrieved. To investigate whether an agent's performance continues to rise
扩展记忆检索提升代理性能。虽然我们的主要实验已证明程序记忆提升了代理的任务准确率并减少了所需步骤数，基于向量的存储和检索相较于人类程序记忆具有优势：它们可以在总容量和检索记忆数量上进行扩展。为探究代理性能是否会随着

as the procedural-memory store and the number of retrieved memories increase, we designed a set of follow-up experiments. As showned in Figure 4 (b), as the number of retrieved procedural memories increases, the agent's performance also improves steadily, exhibiting an upward trend followed by a plateau. However, retrieving too many memories can lead to a decline in the agent's performance. This is because excessive retrieval can affect the context length and also introduce less accurate procedural memories, which can interfere with the overall effectiveness.
程序记忆库规模和检索记忆数量的增加而持续提升，我们设计了一系列后续实验。如图4(b)所示，随着检索的程序记忆数量增加，代理性能稳步提升，呈现先上升后趋于平稳的趋势。然而，检索过多记忆会导致代理性能下降。这是因为过度检索会影响上下文长度，并引入准确度较低的程序记忆，干扰整体效果。

21 Conclusion and Future Work

22 结论与未来工作

We introduce Mem^p ,a task-agnostic framework that elevates procedural memory to a core optimization target in LLM-based agents. By systematically studying strategies for memory construction, retrieval, and updating, Memp enables agents to distill, reuse, and refine their own past experiences across diverse, long-horizon tasks. Empirical results on housework automation and information-seeking benchmarks show that leveraging procedural memory significantly boosts task success rates and efficiency. Beyond improving individual episodes, Mem^p supports continual learning and robust generalization, marking a step toward self-improving, resilient agents.
我们提出了Mem^p，一个任务无关的框架，将程序记忆提升为基于大型语言模型（LLM）代理的核心优化目标。通过系统研究记忆构建、检索和更新策略，Memp使代理能够在多样且长远的任务中提炼、复用和完善自身过往经验。在家务自动化和信息检索基准测试中的实证结果表明，利用程序记忆显著提升了任务成功率和效率。除了改善单次任务表现，Mem^p还支持持续学习和稳健泛化，迈出了实现自我提升、具备韧性的代理的重要一步。

In our experiments, Mem^p has achieved promising results in both construction and retrieval. Moving forward, we plan to enhance this work in several ways. Firstly, we will develop more diverse retrieval strategies. The current approach involves constructing different keys for vector-based retrieval. However, traditional methods like BM25 could also be explored to retrieve precise memories more effectively. Secondly,in Mem^p ,we currently rely on the standard
在我们的实验中，Mem^p在构建和检索方面均取得了可喜成果。未来，我们计划从多个方面增强该工作。首先，将开

发更多样化的检索策略。目前的方法涉及为基于向量的检索构建不同的键，但也可探索传统方法如BM25，以更有效地检索精准记忆。其次，在Mem^p中，我们目前依赖基准测试提供的标准

reward signals provided by the benchmark. However, in real-world scenarios, many tasks do not have clear reward signals, making it difficult for the agent to determine whether a task has been completed successfully. In such cases, using a large language model (LLM) as a judge to assess task completion could be a viable solution. This would transform the agent's lifecycle into a continuous loop of executing tasks, self-assessing completion, building memories, and then proceeding to new tasks.
奖励信号。然而，在现实场景中，许多任务缺乏明确的奖励信号，使代理难以判断任务是否成功完成。在此类情况下，使用大型语言模型（LLM）作为评判者来评估任务完成情况可能是一种可行方案。这将使代理的生命周期转变为一个持续循环：执行任务、自我评估完成度、构建记忆，然后继续新任务。

23 References

24 参考文献

Anthropic. 2022. Claude 3.5 Sonnet System Card.
Anthropic. 2022. Claude 3.5 Sonnet 系统说明。

Barres, V.; Dong, H.; Ray, S.; Si, X.; and Narasimhan, K. 2025. τ^2 -Bench: Evaluating Conversational Agents in a Dual-Control Environment.
Barres, V.; Dong, H.; Ray, S.; Si, X.; 和 Narasimhan, K. 2025. τ^2-Bench：在双控环境中评估对话代理。

Chen, C.; Hao, X.; Liu, W.; Huang, X.; Zeng, X.; Yu, S.; Li, D.; Wang, S.; Gan, W.; Huang, Y.; et al. 2025. ACEBench: Who Wins the Match Point in Tool Usage?
Chen, C.; Hao, X.; Liu, W.; Huang, X.; Zeng, X.; Yu, S.; Li, D.; Wang, S.; Gan, W.; Huang, Y.; 等. 2025. ACEBench：谁将在工具使用中赢得赛点？

Chen, M.; Li, Y.; Yang, Y.; Yu, S.; Lin, B.; and He, X. 2024. AutoManual: Constructing Instruction Manuals by LLM Agents via Interactive Environmental Learning. arXiv:2405.16247.
Chen, M.; Li, Y.; Yang, Y.; Yu, S.; Lin, B.; 和 He, X. 2024. AutoManual：通过交互式环境学习由LLM代理构建说明手册。arXiv:2405.16247。

Chhikara, P.; Khant, D.; Aryan, S.; Singh, T.; and Yadav, D. 2025. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory.
Chhikara, P.; Khant, D.; Aryan, S.; Singh, T.; 和 Yadav, D. 2025. Mem0：构建具备可扩展长期记忆的生产级AI代理。

Cohen, N. J.; and Squire, L. R. 1980. Preserved learning and retention of pattern-analyzing skill in amnesia: Dissociation of knowing how and knowing that.
Cohen, N. J.; 和 Squire, L. R. 1980. 失忆症中模式分析技能的学习与保持：知道如何（knowing how）与知道什么（knowing that）的分离。

Dong, G.; Chen, Y.; Li, X.; Jin, J.; Qian, H.; Zhu, Y.; Mao, H.; Zhou, G.; Dou, Z.; and Wen, J.-R. 2025. Tool-Star: Empowering LLM-Brained Multi-Tool Reasoner via Reinforcement Learning.
Dong, G.; Chen, Y.; Li, X.; Jin, J.; Qian, H.; Zhu, Y.; Mao, H.; Zhou, G.; Dou, Z.; 和 Wen, J.-R. 2025. Tool-Star：通过强化学习赋能具备大型语言模型（LLM）大脑的多工具推理器。

Fang, R.; Wang, X.; Liang, Y.; Qiao, S.; Wu, J.; Xi, Z.; Zhang, N.; Jiang, Y.; Xie, P.; Huang, F.; et al. 2025. SynWorld: Virtual Scenario Synthesis for Agentic Action Knowledge Refinement.
Fang, R.; Wang, X.; Liang, Y.; Qiao, S.; Wu, J.; Xi, Z.; Zhang, N.; Jiang, Y.; Xie, P.; Huang, F.; 等. 2025. SynWorld：用于代理行动知识精炼的虚拟场景合成。

Feng, E.; Zhou, W.; Liu, Z.; Chen, L.; Dong, Y.; Zhang, C.; Zhao, Y.; Du, D.; Hua, Z.; Xia, Y.; et al. 2025. Get Experience from Practice: LLM Agents with Record & Replay.
Feng, E.; Zhou, W.; Liu, Z.; Chen, L.; Dong, Y.; Zhang, C.; Zhao, Y.; Du, D.; Hua, Z.; Xia, Y.; 等. 2025. 从实践中获取经验：具备记录与回放功能的大型语言模型代理。

Gupta, P.; and Cohen, N. J. 2002. Theoretical and computational analysis of skill learning, repetition priming, and procedural memory.
Gupta, P.; 和 Cohen, N. J. 2002. 技能学习、重复启动效应与程序性记忆的理论与计算分析。

Gur, I.; Furuta, H.; Huang, A.; Safdari, M.; Matsuo, Y.; Eck, D.; and Faust, A. 2023. A real-world webagent with planning, long context understanding, and program synthesis.
Gur, I.; Furuta, H.; Huang, A.; Safdari, M.; Matsuo, Y.; Eck, D.; 和 Faust, A. 2023. 具备规划、长上下文理解与程序合成能力的真实世界网络代理。

Hou, Y.; Tamoto, H.; and Miyashita, H. 2024. "my agent understands me better": Integrating dynamic human-like memory recall and consolidation in llm-based agents. In Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, 1-7.
Hou, Y.; Tamoto, H.; 和 Miyashita, H. 2024. “我的代理更懂我”：在基于大型语言模型的代理中整合动态类人记忆回忆与巩固。载于CHI人机交互大会扩展摘要，1-7页。

Hu, M.; Chen, T.; Chen, Q.; Mu, Y.; Shao, W.; and Luo, P. 2024a. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model.
Hu, M.; Chen, T.; Chen, Q.; Mu, Y.; Shao, W.; 和 Luo, P. 2024a. Hiagent：用于解决长远任务的大型语言模型分层工作记忆管理。

Hu, M.; Chen, T.; Chen, Q.; Mu, Y.; Shao, W.; and Luo, P. 2024b. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model.
Hu, M.; Chen, T.; Chen, Q.; Mu, Y.; Shao, W.; 和 Luo, P. 2024b. Hiagent：用于解决长远任务的大型语言模型分层工作记忆管理。

Ifargan, T.; Hafner, L.; Kern, M.; Alcalay, O.; and Kishony, R. 2025. Autonomous llm-driven research—from data to human-verifiable research papers.
Ifargan, T.; Hafner, L.; Kern, M.; Alcalay, O.; 和 Kishony, R. 2025. 自主大型语言模型驱动的研究——从数据到人类可验证的研究论文。

Laird, J. E. 2022. Introduction to the soar cognitive architecture.
Laird, J. E. 2022. SOAR认知架构导论。

Lan, W.; Tang, Z.; Liu, M.; Chen, Q.; Peng, W.; Chen, Y. P.; and Pan, Y. 2025. The large language models on biomedical data analysis: a survey.
Lan, W.; Tang, Z.; Liu, M.; Chen, Q.; Peng, W.; Chen, Y. P.; 和 Pan, Y. 2025. 大型语言模型在生物医学数据分析中的应用综述。

Li, G.; Hammoud, H.; Itani, H.; Khizbullin, D.; and Ghanem, B. 2023. Camel: Communicative agents for" mind" exploration of large language model society.
Li, G.; Hammoud, H.; Itani, H.; Khizbullin, D.; 和 Ghanem, B. 2023. Camel：用于大型语言模型社会“心智”探索的交流代理。

Li, Z.; Song, S.; Xi, C.; Wang, H.; Tang, C.; Niu, S.; Chen, D.; Yang, J.; Li, C.; Yu, Q.; et al. 2025. Memos: A memory os for ai system.
Li, Z.; Song, S.; Xi, C.; Wang, H.; Tang, C.; Niu, S.; Chen, D.; Yang, J.; Li, C.; Yu, Q.; 等. 2025. Memos：AI系统的记忆操作系统。

Liu, B.; Li, X.; Zhang, J.; Wang, J.; He, T.; Hong, S.; Liu, H.; Zhang, S.; Song, K.; Zhu, K.; et al. 2025a. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems.
Liu, B.; Li, X.; Zhang, J.; Wang, J.; He, T.; Hong, S.; Liu, H.; Zhang, S.; Song, K.; Zhu, K.; 等. 2025a. 基础代理的进展与挑战：从脑启发智能到进化、协作与安全系统。

Liu, Y.; Si, C.; Narasimhan, K.; and Yao, S. 2025b. Contextual Experience Replay for Self-Improvement of Language Agents.
Liu, Y.; Si, C.; Narasimhan, K.; 和 Yao, S. 2025b. 用于语言代理自我提升的上下文经验回放。

Liu, Z.; Chai, J.; Zhu, X.; Tang, S.; Ye, R.; Zhang, B.; Bai, L.; and Chen, S. 2025c. Ml-agent: Reinforcing llm agents for autonomous machine learning engineering.
Liu, Z.; Chai, J.; Zhu, X.; Tang, S.; Ye, R.; Zhang, B.; Bai, L.; 和 Chen, S. 2025c. Ml-agent：强化大型语言模型(LLM)代理以实现自主机器学习工程。

Lu, F.; Zhong, Z.; Liu, S.; Fu, C.-W.; and Jia, J. 2025. ARPO: End-to-End Policy Optimization for GUI Agents with Experience Replay.
Lu, F.; Zhong, Z.; Liu, S.; Fu, C.-W.; 和 Jia, J. 2025. ARPO：基于经验回放的GUI代理端到端策略优化。

Luo, R.; Wang, L.; He, W.; and Xia, X. 2025. Gui-r1: A generalist r1-style vision-language action model for gui agents.
Luo, R.; Wang, L.; He, W.; 和 Xia, X. 2025. Gui-r1：一种通用的R1风格视觉-语言动作模型，用于GUI代理。

Mavroudis, V. 2024. LangChain v0. 3.
Mavroudis, V. 2024. LangChain v0.3。

OpenAI. 2022. GPT-4 System Card.
OpenAI. 2022. GPT-4 系统说明。

OpenAI. 2025. Deep Research System Card.
OpenAI. 2025. 深度研究系统说明。

Ou, Y.; Luo, Y.; Zheng, J.; Wei, L.; Qiao, S.; Zhang, J.; Zheng, D.; Chen, H.; and Zhang, N. 2025. AutoMind: Adaptive Knowledgeable Agent for Automated Data Science.
Ou, Y.; Luo, Y.; Zheng, J.; Wei, L.; Qiao, S.; Zhang, J.; Zheng, D.; Chen, H.; 和 Zhang, N. 2025. AutoMind：用于自动化数据科学的自适应知识型代理。

Puterman, M. L. 1990. Markov decision processes.
Puterman, M. L. 1990. 马尔可夫决策过程。

Qiao, S.; Fang, R.; Qiu, Z.; Wang, X.; Zhang, N.; Jiang, Y.; Xie, P.; Huang, F.; and Chen, H. 2025. Benchmarking Agen-tic Workflow Generation. In The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net.
Qiao, S.; Fang, R.; Qiu, Z.; Wang, X.; Zhang, N.; Jiang, Y.; Xie, P.; Huang, F.; 和 Chen, H. 2025. 代理工作流生成基准测试。发表于第十三届国际学习表征会议(ICLR 2025)，新加坡，2025年4月24-28日。OpenReview.net。

Qiao, S.; Ou, Y.; Zhang, N.; Chen, X.; Yao, Y.; Deng, S.; Tan, C.; Huang, F.; and Chen, H. 2023. Reasoning with Language Model Prompting: A Survey. In Rogers, A.; Boyd-Graber, J. L.; and Okazaki, N., eds., Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023, 5368-5393. Association for Computational Linguistics.
Qiao, S.; Ou, Y.; Zhang, N.; Chen, X.; Yao, Y.; Deng, S.; Tan, C.; Huang, F.; 和 Chen, H. 2023. 语言模型提示推理综述。收录于Rogers, A.; Boyd-Graber, J. L.; 和 Okazaki, N. 编，《第61届计算语言学协会年会论文集（第一卷：长文）》，ACL 2023，加拿大多伦多，2023年7月9-14日，5368-5393页。计算语言学协会。

Qin, Y.; Ye, Y.; Fang, J.; Wang, H.; Liang, S.; Tian, S.; Zhang, J.; Li, J.; Li, Y.; Huang, S.; et al. 2025. Ui-tars: Pioneering automated gui interaction with native agents.
Qin, Y.; Ye, Y.; Fang, J.; Wang, H.; Liang, S.; Tian, S.; Zhang, J.; Li, J.; Li, Y.; Huang, S.; 等. 2025. Ui-tars：开创性地使用本地代理实现自动化GUI交互。

Shridhar, M.; Yuan, X.; Côté, M.; Bisk, Y.; Trischler, A.; and Hausknecht, M. J. 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenRe-view.net.
Shridhar, M.; Yuan, X.; Côté, M.; Bisk, Y.; Trischler, A.; 和 Hausknecht, M. J. 2021. ALFWorld：对齐文本与具身环境以实现交互式学习。发表于第九届国际学习表征会议(ICLR 2021)，线上会议，奥地利，2021年5月3-7日。OpenReview.net。

Su, H.; Sun, R.; Yoon, J.; Yin, P.; Yu, T.; and Arık, S. Ö. 2025. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments.
Su, H.; Sun, R.; Yoon, J.; Yin, P.; Yu, T.; 和 Arık, S. Ö. 2025. Learn-by-interact：面向现实环境中自适应代理的数据中心框架。

Sumers, T.; Yao, S.; Narasimhan, K.; and Griffiths, T. 2023a. Cognitive architectures for language agents.
Sumers, T.; Yao, S.; Narasimhan, K.; 和 Griffiths, T. 2023a. 语言代理的认知架构。

Sumers, T.; Yao, S.; Narasimhan, K.; and Griffiths, T. 2023b. Cognitive architectures for language agents.
Sumers, T.; Yao, S.; Narasimhan, K.; 和 Griffiths, T. 2023b. 语言代理的认知架构。

Sun, J.; Zhang, Q.; Duan, Y.; Jiang, X.; Cheng, C.; and Xu, R. 2024. Prompt, plan, perform: Llm-based humanoid control via quantized imitation learning. In 2024 IEEE International Conference on Robotics and Automation (ICRA), 16236-16242. IEEE.
孙杰；张强；段勇；姜晓；程晨；徐睿。2024。提示、规划、执行：基于量化模仿学习的大型语言模型（LLM）人形控制。在2024年IEEE国际机器人与自动化会议（ICRA），16236-16242。IEEE。

Tan, X.; Li, B.; Qiu, X.; Qu, C.; Chu, W.; Xu, Y.; and Qi, Y. 2025. Meta-Agent-Workflow: Streamlining Tool Usage in LLMs through Workflow Construction, Retrieval, and Refinement. In Companion Proceedings of the ACM on Web Conference 2025, WWW '25, 458-467. New York, NY, USA: Association for Computing Machinery. ISBN 9798400713316.
谭翔；李斌；邱翔；曲超；褚伟；徐阳；齐勇。2025。元代理工作流：通过工作流构建、检索与优化简化大型语言模型中的工具使用。在2025年ACM网络会议伴随论文集，WWW '25，458-467。美国纽约：计算机协会。ISBN 9798400713316。

Tang, X.; Qin, T.; Peng, T.; Zhou, Z.; Shao, D.; Du, T.; Wei, X.; Xia, P.; Wu, F.; Zhu, H.; Zhang, G.; Liu, J.; Wang, X.; Hong, S.; Wu, C.; Cheng, H.; Wang, C.; and Zhou, W. 2025. Agent KB: Leveraging Cross-Domain Experience for Agen-tic Problem Solving. arXiv:2507.06229.
唐翔；秦涛；彭涛；周志；邵东；杜涛；魏翔；夏鹏；吴飞；朱浩；张刚；刘杰；王翔；洪松；吴超；程浩；王超；周伟。2025。Agent KB：利用跨领域经验进行代理式问题解决。arXiv:2507.06229。

Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; and Anandkumar, A. 2023. Voyager: An open-ended embodied agent with large language models.
王刚；谢勇；姜毅；Mandlekar, A.；肖晨；朱阳；范磊；Anandkumar, A. 2023。Voyager：基于大型语言模型的开放式具身代理。

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2024a. A survey on large language model based autonomous agents.
王磊；马超；冯翔；张志；杨辉；张健；陈志；唐杰；陈晓；林宇；等。2024a。基于大型语言模型的自主代理综述。

Wang, Z.; Xu, H.; Wang, J.; Zhang, X.; Yan, M.; Zhang, J.; Huang, F.; and Ji, H. 2025. Mobile-agent-e: Self-evolving mobile assistant for complex tasks.
王志；徐浩；王军；张翔；闫明；张健；黄峰；季浩。2025。Mobile-agent-e：面向复杂任务的自我进化移动助手。

Wang, Z. Z.; Mao, J.; Fried, D.; and Neubig, G. 2024b. Agent workflow memory.
王志忠；毛军；Fried, D.；Neubig, G. 2024b。代理工作流记忆。

Wu, J.; Li, B.; Fang, R.; Yin, W.; Zhang, L.; Tao, Z.; Zhang, D.; Xi, Z.; Fu, G.; Jiang, Y.; et al. 2025a. WebDancer: Towards Autonomous Information Seeking Agency.
吴军；李斌；方锐；尹伟；张磊；陶志；张东；席志；傅刚；姜毅；等。2025a。WebDancer：迈向自主信息搜索代理。

Wu, J.; Yin, W.; Jiang, Y.; Wang, Z.; Xi, Z.; Fang, R.; Zhang, L.; He, Y.; Zhou, D.; Xie, P.; and Huang, F. 2025b. Web-Walker: Benchmarking LLMs in Web Traversal. In Che, W.; Nabende, J.; Shutova, E.; and Pilehvar, M. T., eds., Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 10290-10305. Vienna, Austria: Association for Computational Linguistics. ISBN 979-8-89176-251-0.
吴军；尹伟；姜毅；王志；席志；方锐；张磊；何阳；周东；谢鹏；黄峰。2025b。Web-Walker：大型语言模型在网页遍历中的基准测试。收录于Che, W.；Nabende, J.；Shutova, E.；Pilehvar, M. T.编辑，第63届计算语言学协会年会论文集（第一卷：长文），10290-10305。奥地利维也纳：计算语言学协会。ISBN 979-8-89176-251-0。

Wu, X.; Bu, Y.; Cai, Y.; and Wang, T. 2024. Updating Large Language Models' Memories with Time Constraints.
吴翔；卜阳；蔡阳；王涛。2024。带时间约束的大型语言模型记忆更新。

x.ai. 2025. Grok 3 Beta - The Age of Reasoning Agents.
x.ai。2025。Grok 3 Beta——推理代理时代。

Xi, Z.; Chen, W.; Guo, X.; He, W.; Ding, Y.; Hong, B.; Zhang, M.; Wang, J.; Jin, S.; Zhou, E.; et al. 2025. The rise and potential of large language model based agents: A survey.
席志；陈伟；郭翔；何伟；丁毅；洪斌；张明；王军；金森；周恩；等。2025。基于大型语言模型代理的兴起与潜力综述。

Xia, M.; Ruehle, V.; Rajmohan, S.; and Shokri, R. 2025. Minerva: A Programmable Memory Test Benchmark for Language Models.
夏明；Ruehle, V.；Rajmohan, S.；Shokri, R. 2025。Minerva：面向语言模型的可编程记忆测试基准。

Xie, J.; Zhang, K.; Chen, J.; Zhu, T.; Lou, R.; Tian, Y.; Xiao, Y.; and Su, Y. 2024. TravelPlanner: A Benchmark for Real-World Planning with Language Agents. In Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net.
谢军；张凯；陈杰；朱涛；娄然；田阳；肖阳；苏阳。2024。TravelPlanner：面向语言代理的真实世界规划基准。收录于第四十一届国际机器学习大会，ICML 2024，奥地利维也纳，2024年7月21-27日。OpenReview.net。

Xu, W.; Liang, Z.; Mei, K.; Gao, H.; Tan, J.; and Zhang, Y. 2025. A-mem: Agentic memory for llm agents.
徐伟；梁志；梅凯；高浩；谭军；张扬。2025。A-mem：面向大型语言模型代理的代理记忆。

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; et al. 2024a. Qwen2.5 technical report.
杨安；杨斌；张斌；惠斌；郑斌；余斌；李超；刘东；黄峰；魏浩；等。2024a。Qwen2.5技术报告。

Yang, H.; Yue, S.; and He, Y. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions.
杨辉；岳松；何勇。2023。用于在线决策的Auto-gpt：基准测试及附加观点。

Yang, Y.; Zhou, T.; Li, K.; Tao, D.; Li, L.; Shen, L.; He, X.; Jiang, J.; and Shi, Y. 2024b. Embodied multi-modal agent trained by an llm from a parallel textworld. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 26275-26285.
杨洋；周涛；李凯；陶东；李磊；沈磊；何翔；蒋军；石勇。2024b。由大型语言模型（LLM）从平行文本世界训练的具身多模态代理。在IEEE/CVF计算机视觉与模式识别会议论文集，26275-26285页。

Yao, S.; Shinn, N.; Razavi, P.; and Narasimhan, K. R. 2025. τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. In The Thirteenth International Conference on Learning Representations.
姚松；辛恩；拉扎维；纳拉西姆汉，K. R. 2025。τ-bench：真实世界领域中工具-代理-用户交互的基准测试。第十三届国际学习表征会议论文集。

Yu, H.; Chen, T.; Feng, J.; Chen, J.; Dai, W.; Yu, Q.; Zhang, Y.-Q.; Ma, W.-Y.; Liu, J.; Wang, M.; and Zhou, H. 2025. MemAgent: Reshaping Long-Context LLM with Multi-Conv RL-based Memory Agent. arXiv:2507.02259.
余浩；陈涛；冯军；陈杰；戴伟；余强；张永强；马文英；刘军；王明；周宏。2025。MemAgent：基于多卷积强化学习的记忆代理重塑长上下文大型语言模型。arXiv:2507.02259。

Zhang, Z.; Bo, X.; Ma, C.; Li, R.; Chen, X.; Dai, Q.; Zhu, J.; Dong, Z.; and Wen, J.-R. 2024. A survey on the memory mechanism of large language model based agents.
张志；薄翔；马超；李锐；陈晓；戴强；朱军；董志；温建荣。2024。大型语言模型代理的记忆机制综述。

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models.
赵文轩；周康；李军；唐涛；王翔；侯阳；闵阳；张斌；张杰；董志；等。2023。大型语言模型综述。

Zheng, L.; Wang, R.; Wang, X.; and An, B. ???? Synapse: Trajectory-as-Exemplar Prompting with Memory for Computer Control. In The Twelfth International Conference on Learning Representations.
郑磊；王锐；王翔；安斌。???? Synapse：基于记忆的轨迹示例提示用于计算机控制。第十二届国际学习表征会议论文集。

Zhong, W.; Guo, L.; Gao, Q.; Ye, H.; and Wang, Y. 2024a. Memorybank: Enhancing large language models with long-term memory. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 19724-19731.
钟伟；郭磊；高强；叶华；王勇。2024a。Memorybank：利用长期记忆增强大型语言模型。在AAAI人工智能会议论文集，第38卷，19724-19731页。

Zhong, W.; Guo, L.; Gao, Q.; Ye, H.; and Wang, Y. 2024b. Memorybank: Enhancing large language models with long-term memory. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, 19724-19731.
钟伟；郭磊；高强；叶华；王勇。2024b。Memorybank：利用长期记忆增强大型语言模型。在AAAI人工智能会议论文集，第38卷，19724-19731页。

Zhou, W.; Jiang, Y. E.; Li, L.; Wu, J.; Wang, T.; Qiu, S.; Zhang, J.; Chen, J.; Wu, R.; Wang, S.; Zhu, S.; Chen, J.; Zhang, W.; Tang, X.; Zhang, N.; Chen, H.; Cui, P.; and Sachan, M. 2023. Agents: An Open-source Framework for Autonomous Language Agents. arXiv:2309.07870.
周伟；蒋业恩；李磊；吴军；王涛；邱松；张杰；陈杰；吴锐；王松；朱松；陈杰；张伟；唐翔；张楠；陈浩；崔鹏；萨昌。2023。Agents：一个开源的自主语言代理框架。arXiv:2309.07870。

Zhou, W.; Ou, Y.; Ding, S.; Li, L.; Wu, J.; Wang, T.; Chen, J.; Wang, S.; Xu, X.; Zhang, N.; Chen, H.; and Jiang, Y. E. 2024. Symbolic Learning Enables Self-Evolving Agents. arXiv:2406.18532.
周伟；欧阳；丁松；李磊；吴军；王涛；陈杰；王松；徐翔；张楠；陈浩；蒋业恩。2024。符号学习实现自我进化代理。arXiv:2406.18532。

Zhou, Z.; Qu, A.; Wu, Z.; Kim, S.; Prakash, A.; Rus, D.; Zhao, J.; Low, B. K. H.; and Liang, P. P. 2025. MEM1: Learning to Synergize Memory and Reasoning for Efficient Long-Horizon Agents. arXiv:2506.15841.
周志；瞿安；吴志；金成；普拉卡什；鲁斯；赵军；罗伯特 · K. H. · 洛；梁鹏鹏。2025。MEM1：学习协同记忆与推理以实现高效长时程代理。arXiv:2506.15841。