MEMPLANNER: Governing Long-Horizon Agency via Dynamic Memory Management and Planning

Anonymous ACL submission

Abstract

Large Language Model (LLM) agents often struggle with long-horizon tasks, as their limited context windows are easily overwhelmed by irrelevant information. Existing memory-augmented agents typically rely on rigid, predefined workflows, which lack the flexibility to adapt to diverse task demands. To address this, we introduce MemPlanner, an autonomous, plan-driven memory agent designed for strategic information management. Unlike traditional systems, MemPlanner employs a plan-then-execute paradigm: the agent first reasons about an optimal memory strategy tailored to the current context, then autonomously orchestrates a sequence of operations. This process is grounded in a hierarchical memory architecture comprising Core Memory for global context and Topic Memory for fine-grained details. By utilizing a progressive disclosure mechanism via a "header-as-index" format, the agent can scan the memory landscape efficiently and retrieve information gradually, minimizing context redundancy. Optimized through reinforcement learning, MemPlanner learns adaptive policies to distill high-level insights and organize factual knowledge. Extensive experiments confirm that MemPlanner achieves substantial improvements over existing agents, demonstrating superior robustness and strategy adaptation in complex, long-horizon environments.

1 Introduction

Enabling LLM agents to tackle long-horizon, complex tasks has become a central pursuit in the field of artificial intelligence(Chen et al., 2025; Wang et al., 2025c). While LLMs demonstrate remarkable reasoning capabilities, they remain fundamentally constrained by their inherent statelessness and finite context windows(Wang et al., 2024; Fang et al., 2025).To bridge the gap between limited contextual capacity and the continuity required for extended interactions, researchers have actively developed memory-augmented agents. By equipping LLMs with external memory modules—ranging from simple key-value stores to hierarchical databases—these systems aim to persist and retrieve critical information across long timeframes(Packer et al., 2023; Chhikara et al., 2025; Wang and Chen, 2025).

Although existing techniques have achieved notable progress, they still suffer from a fundamental limitation: inflexibility. *(1) Predefined memory workflows.* Current memory-augmented agents predominantly rely on static, expert-crafted workflows or heuristic rules to govern memory updates (Chhikara et al., 2025; Li et al., 2025). For instance, some approaches compulsorily update memory at every step (Yu et al., 2025a), regardless of whether new information is meaningful or necessary. As a result, agents are forced to apply uniform processing logic across diverse tasks and contexts, which constrains their adaptability and generalizability. Ideally, agents should be able to learn on their own when to update memory, rather than following fixed, predefined update schedules. *(2) Static memory strategies.* Existing techniques typically predefine what information should be stored in memory (Wang et al., 2025b). However, such predefined memory representations are inherently limited in their ability to adapt across scenarios. Different tasks demand different types of memory content. For example, in conversational tasks, memory should retain factual information to support answering detailed questions, whereas in classification tasks, memory should capture the decision criteria that distinguish categories. A single, predefined form of memory therefore cannot generalize effectively across diverse task settings.

To address the two limitations discussed above, we introduce two complementary design choices. *(1) Allowing agents to decide when to update memory.* Rather than constraining memory updates within a fixed workflow or forcing models to update

memory at every step, we grant agents the autonomy to determine when memory modification is necessary. To enable this flexibility, we provide a set of memory tools, including inserting, modifying, and deleting memories, which the model can invoke based on its own judgment. *(2) Enabling agents to reason about memory strategy through planning.* We introduce an explicit planning phase that allows the model to reason about what to store in memory. Upon receiving a task description and an incoming information chunk, the model first analyzes the content to formulate an optimal strategy, deciding what information should be preserved in memory. Based on this analysis, it generates a structured memory plan—a high-level directive that guides subsequent actions. The model then executes this plan by orchestrating the available memory tools to perform precise memory storage and organization.

Building on the two designs above, we propose , a plan-driven memory-augmented agent system. In this framework, the model is no longer a passive processor but an autonomous agent that manages its own memory lifecycle. By transitioning to this agentic architecture, our system can independently determine when to trigger memory operations and how to optimize storage through an explicit reasoning-before-execution paradigm. The model first formulates a high-level memory plan tailored to the current task and then executes it through a suite of flexible memory tools. To ground these strategic operations efficiently, we introduce a hierarchical memory architecture. This design balances global context awareness via a compact Core Memory with fine-grained detail storage through a modular Topic Memory. By implementing a progressive disclosure mechanism—where the agent scans lightweight headers before accessing dense content—MemPlanner enables a coarse-to-fine retrieval process. This ensures the model can navigate vast information landscapes with high utility and minimal context redundancy, providing robust support for complex reasoning and decision-making in downstream tasks.

The results of our training-free experiments even perform better than the trained baselines on some tasks (2% improvement on the accurate retrieval task), highlighting the effectiveness of our memory design. We further incorporate agentic reinforcement learning and optimize the system using the GRPO algorithm (Shao et al., 2024). After agentic reinforcement learning, exhibits further performance gains (5.7% improvement after RL), and its memory maintenance becomes more efficient and better aligned with task demands.

In summary, the contributions of this paper are as follows:

- **Plan-Driven Agentic Framework.** We propose MemPlanner, which introduces a "plan-then-execute" paradigm, enabling models to strategically manage their memory lifecycle through autonomous reasoning.
- **Hierarchical Memory Architecture.** We design a dual-layer structure (Core & Topic) that implements progressive disclosure, facilitating efficient, coarse-to-fine retrieval while minimizing context redundancy.
- **Memory-Centric Agentic RL.** We optimize the system via reinforcement learning with specialized memory-driven rewards, significantly enhancing performance in complex, long-horizon tasks.

2 Related Work

2.1 Memory for LLM Agents

Prior research has largely focused on evolving memory architectures to expand storage capacity, progressing from simple retrieval-augmented generation (RAG) to sophisticated hierarchical (Packer et al., 2023) and graph-based structures (Rasmussen et al., 2025; Chhikara et al., 2025). Despite these structural advancements, the strategies governing when and how to update these systems remain predominantly static. Most mechanisms, such as SCM (Wang et al., 2025a) and MemoryBank (Zhong et al., 2024), rely on heuristic workflows or fixed behaviors like continuous fusion, assuming a single interaction pattern suffices for diverse scenarios. While these architectures provide storage depth, their reliance on rigid protocols often obscures fine-grained details and fails to adapt to fluctuating task demands.

To improve flexibility, recent works have shifted toward adaptive management. Approaches like Reflexion (Shinn et al., 2023) and MemAgent (Yu et al., 2025a) leverage linguistic feedback or content optimization to enhance memory relevance. However, these methods are typically restricted by inflexible procedural rules, often enforcing mandatory updates regardless of necessity. This limitation highlights the critical need for a framework that reframes memory management as a holistic,

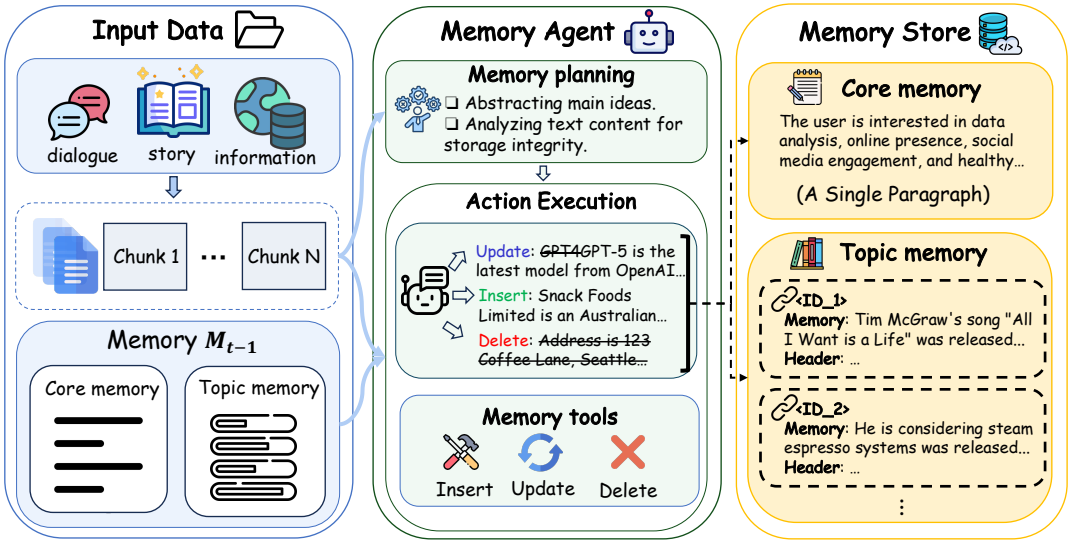

Figure 1: Overview of the MemPlanner framework. The Memory Agent first formulates a high-level memory plan tailored to the current task and then executes it through a suite of flexible memory tools to manage an external memory

autonomous decision-making problem solvable via reinforcement learning.

2.2 Reinforcement Learning for LLMs

Reinforcement learning (RL) has evolved from its initial focus on alignment via RLHF (Ouyang et al., 2022) to a pivotal mechanism for enhancing complex reasoning through verifiable feedback. Recent algorithmic advancements, notably the shift from standard PPO (Schulman et al., 2017) to more efficient variants like GRPO (Shao et al., 2024), have moved the optimization target from learned reward models to ground-truth, outcome-driven signals. While initially validated in deterministic domains like mathematics and logic, this outcome-driven paradigm offers a robust foundation for extending RL to the open-ended, dynamic workflows of autonomous agents.

Directly relevant to our work is the emerging application of RL explicitly for memory construction. Early efforts, such as MEM1 (Zhou et al., 2025) and MemAgent (Yu et al., 2025a), pioneered memory updates but were largely restricted to discrete, localized text modifications. Subsequent frameworks like Memory-R1 (Yan et al., 2025) and Learn-to-Memorize (Zhang et al., 2025) introduced improved representations and tool interfaces. However, these methods predominantly target constrained benchmark settings like LoCoMo (Maharana et al., 2024), which are limited to short contexts (typically under 26k tokens). Furthermore, their reliance on in-distribution training data often creates a generalization gap when facing the unstructured, noise-laden streams characteristic of long-horizon agents. In contrast to prior works optimized for these restricted settings, our framework employs RL to train agents in strategic memory planning, enabling the effective structuring of information across significantly longer horizons.

3 The MemPlanner Framework: Strategic Reasoning and Hierarchical Management

We present MemPlanner, an autonomous agentic framework designed for strategic information management in long-horizon tasks. This section first formalizes the problem setting and introduces our plan-then-execute paradigm. We then describe the hierarchical memory architecture that supports progressive disclosure, followed by the RL strategy used to optimize the agent's decision-making.

3.1 Problem Formulation

We formalize memory construction as a sequential decision-making process where the agent leverages reinforcement learning to evolve adaptive strategies for constructing high-quality memory representations. The agent observes a chunk of user–assistant interaction at each step $t = 1, \ldots, T$. These chunks may contain diverse information types, including casual conversations, task-oriented instructions, and knowledge sharing. Based on the observed chunks, the agent takes a sequence of actions to accomplish the overall goal of memory

construction.

3.2 The Plan-Driven Agentic Framework

Existing techniques often suffer from limited flexibility due to *predefined memory workflows* and *static memory strategies*. To overcome these constraints, we propose a plan-driven memory-augmented agent system, (Figure 1). In this framework, the model is no longer a passive processor but an autonomous agent that manages its own memory lifecycle. By transitioning from a rigid, rule-based approach to an agentic architecture, our system can independently determine when to trigger memory operations (addressing workflow rigidity) and how to optimize storage through an explicit reasoning-before-execution paradigm (addressing strategy limitations).

Plan Generation (Memory Reasoning). Plan Generation (Memory Reasoning). The Plan Generation phase serves as the agent's cognitive reasoning engine, specifically designed to overcome the limitations of "one-size-fits-all" memory usage. Different tasks inherently demand distinct memory strategies: for example, in conversational tasks, the memory should prioritize retaining factual information to support detailed information retrieval; conversely, in classification tasks, the memory should capture the decision criteria that distinguish various categories. Since a single predefined strategy cannot effectively generalize across such diverse settings, our agent first analyzes the semantic content of the task instruction and the incoming interaction c_t to formulate a task-specific memory plan p_t. This plan acts as a strategic blueprint, determining which information is worth preserving and how it should be organized to best serve the task at hand. By formulating this guidance beforehand, the system ensures that its memory usage is not merely reactive, but context-aware and strategy-driven.

Plan Execution (Memory Operations). Once the strategic plan p_t is established, the agent transitions into the role of an active memory controller to implement the directive. Unlike traditional systems that apply fixed updates, our agent treats memory manipulation as a sequential decision-making process. Specifically, we define the agent's observation at step t as $o_t = \{c_t, p_t, M_{t-1}\}$, representing a unified context of the incoming information, the strategic blueprint, and the current internal state. Guided by this observation, the agent autonomously orchestrates a trajectory of memory operations:

$$A_t = a_t^1, \ldots, a_t^{K_t} \tag{1}$$

where each a_t^k is a deliberate action selected from the primitive toolset: $\mathcal{A}_{\text{memory}} = \{\texttt{Insert}, \texttt{Update}, \texttt{Delete}\}$. Rather than treating these as simple labels, the agent invokes them as structured function calls, determining the necessary arguments and parameters based on the evolving context. To maintain logical consistency, the agent executes these actions sequentially to produce a state transition:

$$M_{t-1}^k = \mathcal{T}(M_{t-1}^{k-1}, a_t^k), \quad \text{for } k = 1, \ldots, K_t \tag{2}$$

yielding the final updated memory state $M_t = M_{t-1}^{K_t}$. This iterative execution ensures that the complex strategy formulated during the planning phase is precisely reflected in the final memory state M_n after processing all input chunks $\mathcal{C}$.

3.3 Hierarchical Memory Architecture

To support the diverse operations formulated during the planning phase, we organize the agent's memory into a hierarchical architecture. This design is not merely a storage format but a strategic realization of progressive disclosure, specifically engineered to manifest the agent's plans in the most efficient way possible. By decoupling global awareness from granular details, this hierarchy ensures that the model's memory remains both manageable and high-utility.

Core Memory: Global Context Awareness The Core Memory is designed as a persistent, resident buffer that maintains a compact and global synopsis of the interaction. Its primary role is to provide global context awareness, ensuring that the agent remains grounded in the overall trajectory of the task. By keeping this high-utility summary active within the context window, the agent can maintain a consistent understanding of long-term goals and state transitions without being overwhelmed by raw, unprocessed data.

Topic Memory: Strategic Granularity and Progressive Disclosure To handle fine-grained details and scalable knowledge, the Topic Memory adopts a structured `{Header + Content}` format. This structure is central to our efficiency philosophy: *(1) The Header-as-Index:* The Header serves as a lightweight semantic anchor. It allows the agent to implement progressive disclosure, wherein

it first scans the "memory landscape" through headers rather than loading dense text into the limited context window. *(2) Gradual Retrieval:* This enables a "coarse-to-fine" retrieval process. The agent can efficiently identify and locate the most relevant information gradually—starting with header-based relevance and only "disclosing" the detailed Content when the current plan necessitates deep-dive reasoning.

This hierarchical design achieves maximal efficiency by balancing memory compactness with informational completeness. The dual-layer approach—combining a lightweight resident buffer for global awareness with an expandable topic-indexed repository for modular details—prevents context clutter and reduces redundancy. Ultimately, this ensures that the storage strategies formulated in the planning phase are executed with high retrieval utility and minimal computational overhead.

3.4 Memory Optimization via Reinforcement Learning

We employ **Group Relative Policy Optimization (GRPO)** to train the MemPlanner. For each input sequence $\mathcal{C}$, we sample a group of G trajectories from the policy π_θ. Drawing inspiration from Mem-α (Wang et al., 2025b), we incorporate two distinct categories of reward functions.

The first component is R_{corr}, which serves as a task-success metric reflecting the comprehensiveness of the final memory state M_n. Specifically, given a set of evaluation queries $\mathcal{Q} = \{q_1, \ldots, q_m\}$, we use an external evaluator to answer each query based solely on the constructed memory state M_n, yielding a set of predicted answers $\text{Ans} = \{ans_1, \ldots, ans_m\}$. We compute R_{corr} as: $R_{\text{corr}} = \frac{1}{m}\sum_{i=1}^{m} I[\text{metric}(ans_i, r_i)]$, where $\mathcal{R} = \{r_1, \ldots, r_m\}$ denotes the set of ground-truth answers corresponding to $\mathcal{Q}$, $\text{metric}(\cdot,\cdot)$ is the dataset-specific evaluation metric and $I[\cdot]$ is the indicator function that maps the metric outcome to a reward signal. This result-oriented reward ensures the agent is strictly incentivized to capture and retain salient details necessary for subsequent reasoning.

The second is R_{comp}, defined as a parsimony score that promotes concise representations. We compute this reward as the complement of the retention ratio:$R_{\text{comp}} = 1 - \frac{l_{\text{mem}}}{l_{\text{c}}}$ where l_{c} denotes the total number of tokens in all input text chunks, and l_{mem} denotes the total number of tokens retained in the constructed memory. A higher R_{comp} indicates more aggressive compression. By integrating this objective, we incentivize the agent to identify and discard redundant tokens, ensuring that the constructed memory remains lightweight without sacrificing informational completeness.

We synthesize these distinct metrics into a unified learning signal for each sampled trajectory τ_k. The final reward is formulated as a weighted linear combination: $R(\tau_k) = R_{\text{corr}}(\tau_k) + \beta R_{\text{comp}}(\tau_k)$ where β is a hyperparameter governing the relative importance of structural and compression constraints. We set β to 0.05, following the setting adopted in Mem-α (Wang et al., 2025b).

For a group of G trajectories $\{\tau_1, \ldots, \tau_G\}$ sampled from the same input, the advantage $A(\tau_k)$ is formulated by normalizing the total reward against the group's statistics:

$$A(\tau_k) = \frac{R(\tau_k) - \text{mean}(\{R(\tau_j)\}_{j=1}^{G})}{\text{std}(\{R(\tau_j)\}_{j=1}^{G}) + \epsilon} \quad (3)$$

Crucially, unlike standard proximal policy methods, we purposefully omit the KL divergence regularization term from the objective function (Yu et al., 2025b). This relaxation relieves the agent from the constraint of strictly adhering to the reference policy, thereby broadening the search space for optimal memory strategies. Finally, we optimize the policy using the following objective function:

$$\mathcal{J}(\theta) = E_\tau\left[\frac{1}{|\tau|}\sum_{i=1}^{|\tau|}\min\Big(\rho_i(\theta)A(\tau), \text{clip}(\rho_i(\theta), 1-\varepsilon, 1+\varepsilon)A(\tau)\Big)\right] \quad (4)$$

Where $\rho_i(\theta)$ signifies the importance sampling ratio associated with the i-th sample. This simplified objective fully empowers the model to prioritize trajectories with high relative advantages, fostering the emergence of novel behaviors.

4 EXPERIMENTS

4.1 Experimental Setup

Baselines. We compare MemPlanner with five representative baselines spanning long-context inference, retrieval-based memory, and memory-centric agents: (1) Qwen3-32B: We directly prompt Qwen3-32B (Yang et al., 2025) using its full 32k context window, concatenating the available history as input. (2) RAG: We treat the question as the retrieval query and apply BM25 to rank the

Method	Accurate Retrieval		Test-Time Learning					LRU	Avg.
	R-QA1	R-QA2	TREC-C	NLU	TREC-F	C150	B77	IB-Sum	
Long-Context	28.0	27.0	64.0	74.0	34.0	**86.0**	77.0	12.5	46.1
RAG-Top2	69.0	45.0	69.0	65.0	21.0	70.0	75.0	6.5	50.2
MemAgent	7.0	16.0	37.0	26.0	21.0	25.0	37.0	4.3	19.8
MEM1	7.0	18.0	18.0	0.0	0.0	9.0	0.0	2.9	7.1
Mem-α	74.0	68.0	71.0	71.0	**41.0**	73.0	70.0	**12.9**	59.2
MemPlanner	75.0	70.0	63.0	72.0	33.0	70.0	64.0	8.0	56.8
MemPlanner-RL	**83.0**	**75.0**	**75.0**	**75.0**	39.0	**86.0**	**82.0**	12.7	**65.9**

Table 1: Main results on MemoryAgentBench. The best results are in bold; LRU denotes long-range understanding.

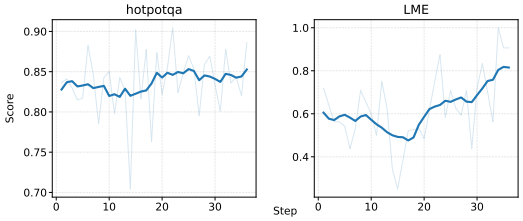

Figure 2: Training curves showing performance during optimization on HotpotQA and LongMemoryEval.

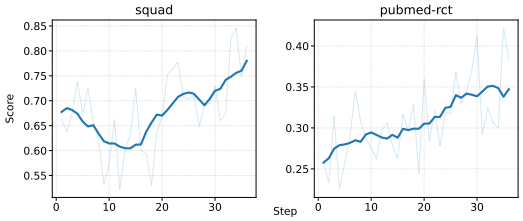

Figure 3: Training curves showing performance during optimization on SQuAD and PubMed-RCT.

Method	R-QA1	R-QA2	TREC-C	IB-Sum
MemPlanner-RL	**83.0**	**75.0**	**75.0**	**12.7**
Planning stage				
w/o Plan	76.0	73.0	71.0	12.4
Memory components				
w/o Core memory	82.0	75.0	55.0	12.5
w/o Topic memory	7.0	17.0	56.0	4.1

Table 2: Ablation study of the planning stage and memory components. R-QA1, R-QA2, TREC-C, and IB-Sum denote Ruler-QA1, Ruler-QA2, TREC-Coarse, and InfBench-Sum, respectively.

historical chunks, selecting the top-2 chunks as the retrieved memory evidence for answering. (3) MemAgent (Yu et al., 2025a): We provide the agent with the task description and the complete set of chunks, allowing it to sequentially review the history and form an internal summary/state before producing the final answer to the question. (4) MEM1 (Zhou et al., 2025): The agent maintains an explicit textual memory buffer (a single paragraph). It iterates over the chunks by repeatedly retrieving relevant pieces, updating the memory buffer accordingly, and then generates the answer conditioned on the final memory state. (5) Mem-α (Wang et al., 2025b): The agent processes chunks in chronological order and incrementally maintains a structured memory composed of three complementary components, which are used jointly to support the final response.

Evaluation Benchmarks. Following Mem-α (Wang et al., 2025b), we employ MemoryAgentBench (Hu et al., 2025) as our benchmark. Adhering to the established experimental setup, we evaluate our method across three dimensions to validate its core competencies. *(1) Accurate Retrieval:* This category evaluates a model's ability to retrieve the correct evidence snippet in response to a query. Retrieval may be single-hop or multi-hop; the latter is employed when the necessary information cannot be obtained through a single request. We measure the agent's ability to precisely locate and recall specific information using the RULER-QA1 (Hsieh et al., 2024) and RULER-QA2 benchmarks. Specifically, RULER-QA1 focuses on questions whose answers can be supported by a directly retrievable span of evidence, whereas RULER-QA2 typically requires aggregating evidence across multiple memory chunks and composing a coherent answer from those pieces. *(2) Test-Time Learning:* We evaluate the agent's capability to acquire task-specific behaviors at deployment time—i.e., adapting to previously unseen classification rules without any parameter updates—by learning from in-context labeled examples. Concretely, we use five multi-class classification benchmarks (TREC-C, TREC-F, NLU, CLINIC, and BANKING77) that span varying label granulari-

ties and domains. During evaluation, the context provides a large set of labeled demonstrations, and the agent is required to infer the label taxonomy from these examples and generalize the induced decision boundaries to new instances. *(3) Long-Range Understanding:* We assess an agent's ability to maintain a global view and synthesize information scattered throughout extremely long contexts (typically $\geq$100k tokens) using the InfBench-Sum summarization task from InfBench (Zhang et al., 2024). In this setting, the input is a long-form narrative that spans many chunks, and the agent must produce a coherent, high-level summary that reflects the overall plot and key developments rather than relying on localized evidence. This evaluation stresses cross-chunk integration and consistency under long-context memory representations.

Implementation Details. We use Qwen3-4B as the default backbone for our method and disable its thinking mode to ensure a fair comparison with the baselines. Building upon existing works (Wang et al., 2025b), we use Qwen3-32B with thinking mode disabled for answer generation, leveraging the memories we have saved. Training consists of 36 steps of GRPO with batch size 24, $\beta = 0.05$, rollout size n = 4 and a max response length of 2048. The training framework is based on VeRL (Sheng et al., 2025). All experiments are conducted on 3 NVIDIA H20 GPUs. For the training set, we adopt the collection from Mem-α (Wang et al., 2025b), which comprises a diverse dataset spanning the three aforementioned task categories, with further details provided in Appendix A.2.

4.2 Main Results

The main experimental results are summarized in Table 1, from which we draw the following observations. **(1) Autonomous, plan-driven memory agents typically outperform fixed workflow-based pipelines.** MemPlanner achieves consistently strong performance across all tasks. For example, on the accurate retrieval benchmarks, MemPlanner attains success rates of 75.0% on RULER-QA1 and 70.0% on RULER-QA2 in a training-free setting, outperforming the reinforcement-learning-trained Mem-α on both the single-hop and multi-hop variants, even without any additional training. These results suggest that explicit planning helps the agent identify and retrieve the most relevant information from memory. By planning ahead, MemPlanner maintains a clearer understanding of what information will be needed later, which enables it to store more relevant content and reduce information overload, remaining robust as the memory environment becomes noisier.

(2) RL training further improves the agent's memory policy, leading to additional gains in performance. Compared with the base model, reinforcement learning yields consistent improvements: after training, the agent achieves an average gain of nearly 10% across all tasks. This result further underscores the importance of making appropriate memory storage decisions for effective retrieval and long-context reasoning. With reward-driven optimization, the agent becomes more selective about what to store, favoring task-relevant content and thereby translating improved memory construction into additional performance gains.

4.3 Analysis of Training Dynamics

Figure 2 and Figure 3 illustrate the training dynamics, reporting the evaluation scores on different datasets over training steps. As shown, the overall trend steadily increases, suggesting that the policy is being progressively optimized. This improvement reflects the agent's growing ability to manage memory via the memory tool in a more effective manner. Moreover, these dynamics underscore the importance of task-level feedback in optimizing memory decisions for effective long-context reasoning. In particular, reinforcement learning enables the agent to learn how to apply memory actions under noisy and extended-context settings, thereby substantially improving final performance.

4.4 Ablation Studies

We conduct ablation studies in Table 4 to quantify the contribution of each component. **(1) Importance of planning:** Removing the planning stage results in consistent performance drops across all tasks, highlighting the central role of planning in accommodating diverse task requirements. In particular, planning helps the agent construct memory more effectively by organizing what to store in a task-aware manner. **(2) Complementary memory components:** We find that removing either Core Memory or Topic Memory degrades performance on specific tasks. Notably, removing Topic Memory causes a catastrophic failure on retrieval benchmarks (an average drop of $\sim$60 percentage points), indicating that the in-context window alone is insufficient to preserve fine-grained details and that external memory with on-demand retrieval is necessary. For classification tasks, ablating either

Method	Accurate Retrieval		Test-Time Learning					LRU	Avg.
	R-197K	R-492K	TREC-C	NLU	TREC-F	C150	B77	IB-Sum	
Long-Context	33K	33K	33K	33K	33K	33K	33K	33K	33K
RAG-Top2	217K	474K	124K	134K	126K	131K	128K	181K	207K
MemAgent	1.02K	1.02K	1.02K	1.02K	0.77K	1.02K	1.02K	0.73K	0.92K
MEM1	0.30K	0.38K	0.16K	0.11K	0.13K	0.28K	0.11K	0.19K	0.21K
Mem-α	160K	323K	120K	142K	123K	18K	133K	19K	129K
MemPlanner-RL	**174K**	**410K**	**123K**	**97K**	**151K**	**38K**	**129K**	**27K**	**144K**

Table 3: Memory tokens on MemoryAgentBench. LRU denotes long-range understanding.

Method	R-QA1	R-QA2	TREC-C	IB-Sum
MemPlanner-RL	**83.0**	**75.0**	**75.0**	**12.7**
Planning stage				
w/o Plan	76.0	73.0	71.0	12.4
Memory components				
w/o Core mem.	82.0	75.0	55.0	12.5
w/o Topic mem.	7.0	17.0	56.0	4.1

Table 4: Ablation study of the planning stage and memory components. R-QA1, R-QA2, TREC-C, and IB-Sum denote Ruler-QA1, Ruler-QA2, TREC-Coarse, and InfBench-Sum, respectively.

Core Memory or Topic Memory reduces accuracy by nearly 20 percentage points, suggesting that the agent relies on the complementarity of these two memory structures to form a complete understanding of a new task. Overall, these results imply that an effective memory architecture should combine short-term and long-term memory, where the two components jointly support efficient storage and strong downstream performance.

4.5 Memory Compression Analysis

We compute the memory token counts for different baselines, as shown in Table 3. Based on these statistics, we examine MemPlanner's advantages in memory utilization and efficiency.

(1) Memory footprint reduction: Compared with chunk-level baselines such as RAG-Top2, which store full retrieved passages, our method substantially reduces the number of stored tokens, lowering the memory budget by roughly 31%. This highlights MemPlanner's efficiency in retaining only the most salient information.

(2) Benefits of structured memory: The comparatively weak results of flat-memory baselines (MEM1 and MemAgent), which compress all information into a single paragraph, suggest that unstructured representations are insufficient for integrating and retrieving diverse evidence in complex tasks. This gap supports the necessity of a hierarchical memory organization, and further indicates that reinforcement learning helps optimize storage decisions under token-budget constraints.

(3) Soundness of the memory components: The results show that, with a memory token budget comparable to Mem-α, our method achieves better performance on most tasks, supporting the effectiveness of our memory design. We argue that memory architectures need not be overly complex; a lighter-weight set of memory components can make memory management more tractable for the agent and improve overall reliability.

5 Conclusion

In this paper, we propose MemPlanner, an autonomous and plan-driven memory agent for long-horizon LLM-based tasks. Unlike conventional memory-augmented agents that rely on rigid retrieval pipelines or fixed access rules, MemPlanner adopts a plan-then-execute paradigm for memory management. Given a task, it first constructs an explicit memory access plan and then progressively retrieves, updates, and integrates relevant information. This enables the agent to reason about what information is needed, when to access it, and how to coordinate different memory components in complex, information-dense environments.

Furthermore, MemPlanner is optimized through reinforcement learning to improve its planning and memory-access decisions. Instead of treating memory retrieval as a static preprocessing step, the agent learns to use memory strategically according to task requirements and intermediate reasoning states. Experimental results demonstrate its robustness and adaptability across challenging long-context scenarios, suggesting that plan-driven memory management is a promising direction for building more efficient, flexible, and reliable LLM agents.

Limitations

Although MemPlanner improves the flexibility of long-horizon memory agents, several limitations remain. First, plan-driven memory management relies on the agent's ability to generate reliable memory plans, and inaccurate planning may lead to incomplete storage or unnecessary memory operations. Second, the hierarchical Core and Topic Memory design introduces a trade-off between compactness and evidence completeness, especially when fine-grained details are distributed across many topics. In addition, reinforcement learning improves memory policies but may require task-specific reward design and additional training cost. Current evaluation mainly focuses on MemoryAgentBench and representative long-context tasks, while broader validation in more interactive, open-ended, and real-world agent environments remains future work.

Ethical Considerations

This work studies memory management mechanisms for long-horizon LLM agents and mainly relies on existing public benchmarks and datasets. We do not introduce new user data or intentionally collect sensitive personal information. However, memory agents are designed to store and reuse historical information, which may raise privacy risks in real-world deployment if user-related content is retained without proper control. Therefore, MemPlanner should be deployed with safeguards such as data filtering, privacy protection, transparent memory editing, and user control over stored memories. In addition, because stronger memory management may improve the efficiency of agentic workflows, appropriate monitoring and governance are necessary to reduce potential misuse.

References

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. 2025. Reinforcement learning for long-horizon interactive llm agents. *arXiv preprint arXiv:2502.01600*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Franck Dernoncourt and Ji-Young Lee. 2017. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 308–313.

Yiming Du, Hongru Wang, Zhengyi Zhao, Bin Liang, Baojun Wang, Wanjun Zhong, Zezhong Wang, and Kam-Fai Wong. 2024. Perltqa: A personal long-term memory dataset for memory classification, retrieval, and fusion in question answering. In *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*, pages 152–164.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, and 1 others. 2025. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2025. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, and 1 others. 2025. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: Towards llms as operating systems.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Zhenhe Wu, ShuangZhi Wu, Zejun Ma, and Zhoujun Li. 2025a. Scm: Enhancing large language model with self-controlled memory framework. In *International Conference on Database Systems for Advanced Applications*, pages 188–203. Springer.

Yu Wang and Xi Chen. 2025. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*.

Yu Wang, Chi Han, Tongtong Wu, Xiaoxin He, Wangchunshu Zhou, Nafis Sadeq, Xiusi Chen, Zexue He, Wei Wang, Gholamreza Haffari, and 1 others. 2024. Towards lifespan cognitive systems. *arXiv preprint arXiv:2409.13265*.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. 2025b. Mem-{\alpha}: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, and 1 others. 2025c. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, and 1 others. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and 1 others. 2025a. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025b. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and 1 others. 2024. Bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277.

Zeyu Zhang, Quanyu Dai, Xu Chen, Rui Li, Zhongyang Li, and Zhenhua Dong. 2025. Memengine: A unified and modular library for developing advanced memory of llm-based agents. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 821–824.

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.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2025. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*.

A Implementation Details

A.1 Training Hyperparameters

We train MemPlanner agents with full parameter optimization. We show some important training hyperparameters in Table 5.

A.2 Training Data

We follow MemoryAgentBench and Mem-α. collected a diverse training dataset spanning three task categories to instill comprehensive agent capabilities:

- **SQuAD** (Rajpurkar et al., 2016), a QA dataset that we adapt for memory training by bundling multiple documents into a single instance, requiring the agent to retain the combined content and later answer questions via precise evidence retrieval.
- **HotPotQA** (Yang et al., 2018), a multi-document, multi-hop QA dataset used to train the agent to follow evidence across sequential chunks, link related documents, and synthesize answers from distributed information.
- **PerLTQA** (Du et al., 2024), a personalization-oriented QA dataset where memory mixes episodic and semantic user information, encouraging relevant memory selection, cross-memory integration, and profile-consistent multi-hop reasoning.
- **LongMemEval** (Wu et al., 2024), a training subset derived from LongMemEval with no overlap with our evaluation split, pairing long "haystack" dialogue contexts (about 10K–30K tokens) with multiple questions to stress long-context retrieval.
- **PubMed-RCT** (Dernoncourt and Lee, 2017), adapted into a test-time learning style sentence-classification task by chunking labeled abstract sentences into conversational segments and mapping semantic roles to numeric IDs (0–4) to emphasize pattern learning.
- **NLU** (Hu et al., 2025), a label-rich sentence classification dataset with 68 classes that we re-chunk into manageable segments while preserving labels and ensuring comprehensive label coverage within each constructed instance.
- **TREC-C** (Hu et al., 2025), a question classification dataset with 6 classes that we similarly re-chunk from very long instances into smaller segments while keeping all labels and maintaining full class coverage per constructed instance.
- **BookSum** (Zhang et al., 2024), a chapter-level summarization dataset (cleaned version) where each chapter is streamed in 10–20 chunks to mimic incremental reading, training the agent to produce summaries aligned with reference content.

Table 5: Training details of MemPlanner.

Parameter	Value
RL Algorithm	GRPO
Base Model	Qwen3-4B
Learning Rate	1e-6
Train Batch Size	24
Number of Rollout	4
Rollout Temperature	1.0
KL Loss Coefficient	0
Clip Ratio	0.2
Training Framework	Verl