Fine-Mem: Fine-Grained Feedback Alignment for Long-Horizon Memory Management

Weitao Ma[1,2*] **Xiaocheng Feng**[1,3†] **Lei Huang**[1] **Xiachong Feng**[4]
Zhanyu Ma[2] **Jun Xu**[2†] **Jiuchong Gao**[2†] **Jinghua Hao**[2] **Renqing He**[2] **Bing Qin**[1,3†]
[1] Harbin Institute of Technology, [2] Meituan
[3] Peng Cheng Laboratory, [4] The University of Hong Kong
{wtma,xcfeng}@ir.hit.edu.cn

Abstract

Effective memory management is essential for large language model agents to navigate long-horizon tasks. Recent research has explored using Reinforcement Learning to develop specialized memory manager agents. However, existing approaches rely on final task performance as the primary reward, which results in severe reward sparsity and ineffective credit assignment, providing insufficient guidance for individual memory operations. To this end, we propose Fine-Mem, a unified framework designed for fine-grained feedback alignment. First, we introduce a Chunk-level Step Reward to provide immediate step-level supervision via auxiliary chunk-specific question answering tasks. Second, we devise Evidence-Anchored Reward Attribution to redistribute global rewards by anchoring credit to key memory operations, based on the specific memory items utilized as evidence in reasoning. Together, these components enable stable policy optimization and align local memory operations with the long-term utility of memory. Experiments on Memalpha and MemoryAgentBench demonstrate that Fine-Mem consistently outperforms strong baselines, achieving superior success rates across various sub-tasks. Further analysis reveals its adaptability and strong generalization capabilities across diverse model configurations and backbones [1].

1 Introduction

Large Language Model (LLM) agents have emerged as a powerful paradigm for addressing various downstream tasks, ranging from multi-turn dialogue to complex multi-step reasoning (Achiam et al., 2023; Yang et al., 2025; Luo et al., 2025). While LLM agents excel in short-term tasks, they struggle with long-horizon scenarios, which involve evolving goals and information tracking over sessions, far exceeding the capacity of fixed context windows (Huang et al., 2023; Liu et al., 2025). Consequently, robust memory systems are essential to mitigate information decay and ensure the long-term coherence of decision-making (Zhang et al., 2025a; Wang et al., 2024; Liang et al., 2025).

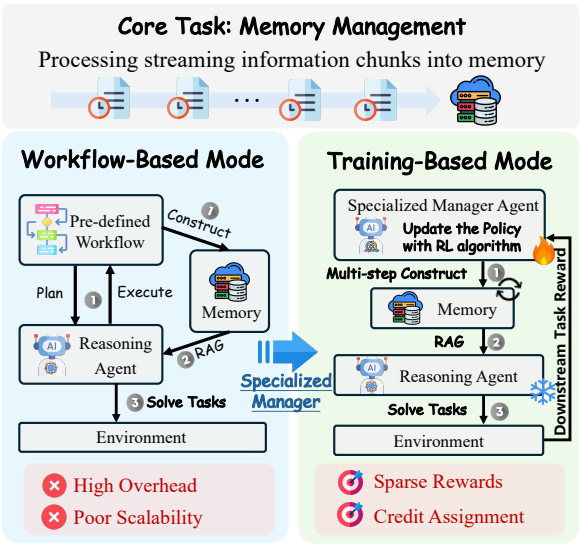

Figure 1: Comparison of memory management paradigms. Left: *Workflow-based mode* relies on pre-defined pipelines and strong LLMs, suffering from high overhead and poor scalability. Right: *Training-based mode* utilizes a specialized manager optimized via RL, improving effectiveness but hindered by sparse rewards and ineffective credit assignment problems.

To bridge this gap, research has shifted from static retrieval to dynamic memory management, focusing on the efficient processing of streaming information chunks into structured memory (Wang and Chen, 2025; Chhikara et al., 2025). As illustrated in Figure 1, current approaches primarily diverge into two paradigms: the *workflow-based mode* and the *training-based mode*. Workflow-based approaches (Xu et al., 2025; Fang et al., 2025), rely on pre-defined heuristic pipelines and large-scale general LLMs. Consequently, they of-

[*] Work done during internship at Meituan.
[†] Corresponding Author.
[1] https://github.com/LVYUERLVR/Fine-Mem

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 19666–19684
July 2-7, 2026

ten suffer from high computational overhead and poor scalability (Hu et al., 2025a). To mitigate these limitations, recent approaches have been trending towards training-based methods, employing specialized, smaller-scale manager agents optimized through Reinforcement Learning (RL) (Yan et al., 2025; Yu et al., 2025). These agents are typically trained to either fold information in working memory (Zhou et al., 2025) or manage long-term memory with sequential operations (Wang et al., 2025). However, relying solely on downstream task performance results in **sparse rewards** and **ineffective credit assignment**, ultimately undermining the manager's performance in complex tasks.

In this work, we present Fine-Mem, a unified reinforcement learning framework that aligns fine-grained feedback with memory operations. To tackle the critical challenges of reward sparsity and ineffective credit assignment, Fine-Mem introduces two components providing dense supervision and principled reward attribution. Firstly, we propose a **Chunk-level Step Reward** (CSR) that provides process supervision via chunk-specific question answering. By using constructed questions to assess the information retention of individual chunks, CSR yields immediate feedback to mitigate sparse global rewards. Secondly, we introduce **Evidence-Anchored Reward Attribution** (EARA) to achieve fine-grained credit assignment. By tracing memory items retrieved in downstream tasks back to critical memory operations, EARA maps final utility to specific steps, effectively redistributing the global reward. Together, these components enable stable policy optimization through Group Relative Policy Optimization, empowering the memory manager to master complex strategies.

To validate Fine-Mem, we conduct extensive experiments on two representative benchmarks, Memalpha and MemoryAgentBench, which assess long-horizon and cross-session memory capabilities. Empirical results demonstrate that Fine-Mem consistently outperforms seven competitive baselines, achieving average improvements of 4.4% on Memalpha and 7.2% on MemoryAgentBench. Moreover, it attains leading performance across all sub-tasks, including Accurate Retrieval, Test-Time Learning, and Long-Range Understanding, while preserving relatively concise memory lengths. Ablation studies confirm that CSR and EARA jointly enhance task performance while effectively constraining memory length. Furthermore, extensive evaluations across varied reasoning models and manager backbones highlight the framework's strong generalization ability in diverse settings.

2 Preliminaries

2.1 Task Formulation

The task of memory management can be formulated as a sequential decision-making process over a stream of historical information, denoted as $\mathcal{C} = \{c_1, c_2, \ldots, c_T\}$, where each c_t represents a history chunk (e.g., a segment of past dialogue turns). Specifically, a "chunk" is defined as a discrete unit of information within the continuous input stream that represents the granularity of input processing, rather than a retrieval unit typically discussed in standard Retrieval-Augmented Generation (RAG) contexts. Our framework consists of two primary agents: a learnable *Memory Manager* π_θ and a fixed *Reasoning Agent* π_{reason}.

At each time step t, the *Memory Manager* π_θ observes the current history chunk c_t and the previous memory state $\mathcal{M}_{t-1}$, generating a set of operations P_t to update the memory:

$$P_t \sim \pi_\theta(\cdot \mid c_t, \mathcal{M}_{t-1}), \quad \mathcal{M}_t = \mathcal{T}(\mathcal{M}_{t-1}, P_t), \tag{1}$$

where $\mathcal{T}$ denotes the state transition function that executes the set of operations P_t. The memory state $\mathcal{M}_t$ can be viewed as a collection of memory items accumulated up to step t.

After processing the entire streaming chunks, the finalized memory state $\mathcal{M}_T = \{m_1, m_2, \ldots\}$ serves as the knowledge foundation for downstream tasks. Given a specific query q_j, the *Reasoning Agent* π_{reason} first retrieves a relevant subset of memory items and then generates an answer a_j conditioned on both the query and the items:

$$M_j = \text{Retrieve}(q_j, \mathcal{M}_T), a_j \sim \pi_{\text{reason}}(\cdot \mid q_j, M_j) \tag{2}$$

where $M_j \subseteq \mathcal{M}_T$. This process simulates long-horizon scenarios, where the agent must rely on the accumulated memory to address complex queries.

2.2 Memory Architecture and Operations

Memory Architecture We adopt a unified single-layer memory architecture to free the *Memory Manager* from deciding where information should be stored. Specifically, each memory item $m_i \in \mathcal{M}$ is represented as a triplet $\{id_i, content_i, s_i\}$, corresponding to a unique identifier, the stored content, and the specific update step, respectively. More

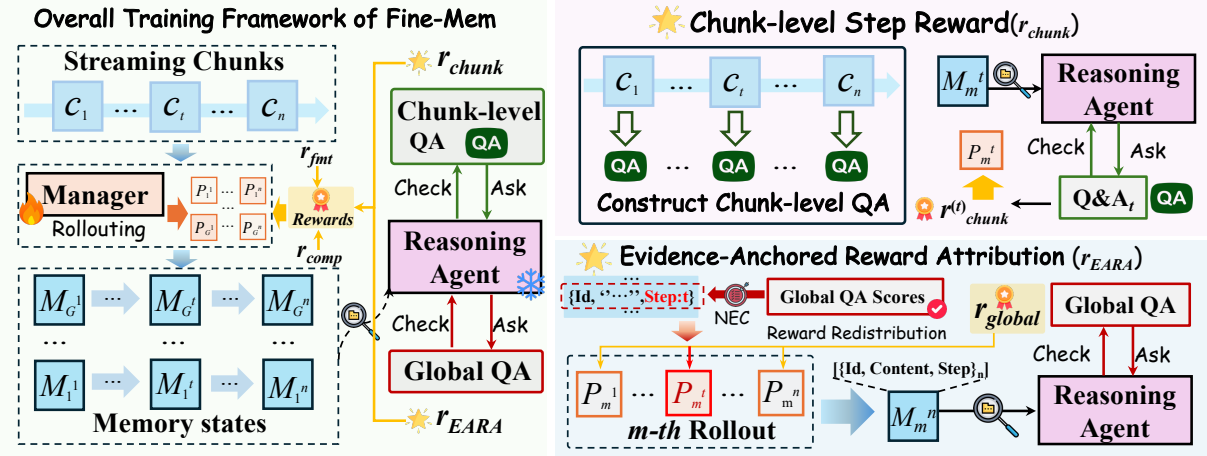

Figure 2: An overview of Fine-Mem. Left: The overall training framework. Right: Two core components designed to enhance training: (1) Chunk-level Step Reward (§3.1), which addresses reward sparsity by generating chunk-level QA tasks to provide step-level feedback for memory operations; (2) Evidence-Anchored Reward Attribution (§3.2), which resolves the credit assignment challenge by redistributing global rewards back to specific rollout steps.

complex multi-layer memory structures would impose substantial challenges on the *Memory Manager*, as they typically require carefully designed supervision or reward signals to guide memory placement. However, it isn't easy to specify in practice and may even degrade the manager's effectiveness when misaligned.

Operation Space Following prior work (Wang et al., 2025; Yan et al., 2025), the *Memory Manager* operates over a compact operation space consisting of four atomic operations: **INSERT**, **UPDATE**, **DELETE**, and **SKIP**. This minimal set affords flexible memory manipulation by enabling addition, refinement, removal, and selective retention of information, while allowing complex behaviors to emerge through the composition of these simple primitives. A more detailed definition of the operations is provided in Appendix A.

3 Methodology

In this section, we present Fine-Mem, a unified reinforcement learning framework for training the memory manager with fine-grained reward assignment, as illustrated in Figure 2. Fine-Mem comprises two key components: (1) **Chunk-level Step Reward**, which introduces chunk-level QA to provide richer step-level supervision for operations, and (2) **Evidence-Anchored Reward Attribution**, which redistributes global rewards back to individual rollout steps for more precise credit assignment.

3.1 Chunk-level Step Reward

Current reinforcement learning methods for memory managers rely primarily on sparse, rollout-level rewards, providing no direct supervision for individual decision steps. However, unlike traditional long-horizon reasoning tasks (e.g., mathematical reasoning or code generation), memory management can be viewed as a sequence of locally grounded decisions triggered by incoming chunks, where each step mainly depends on the current observation and the existing memory state.

An ideal reward for each step-level operation set should jointly consider two aspects: the quality of storing information within the current chunk, and its influence on the handling of information from other chunks. Accordingly, we introduce chunk-level step rewards (r_{chunk}), which are computed on the chunk-level QA pairs to model local information processing, while retaining global QA accuracy to capture cross-chunk effects.

The chunk-level QA pairs are generated by first extracting concise, factoid-style questions from each chunk using `GPT-4o-mini`, with paraphrased formulations included to enhance generalization. To ensure that each QA pair is fully grounded in the chunk, a verifier model answers each question using only the corresponding chunk content, and any pair that cannot be correctly answered is discarded. After removing duplicates, a fixed budget of five QA pairs is retained per chunk. The complete construction process is detailed in the Appendix B.1. Together, this dual QA accuracy design encourages precise acquisition of new information while ensur-

Algorithm 1 Evidence-Anchored Reward Attribution for a Single Rollout

Input: Global QA scores $\{s_j\}_{j=1}^{n}$, retrieved memory sets $\{M_j\}$, memory-to-step map $\phi(\cdot)$, global reward r_{global}, factor β

Output: Step-level rewards $\{r_{EARA}^{(t)}\}_{t=1}^{T}$

Initialize evidence contributions $N_t \leftarrow 0 \quad \forall t$
for $j = 1$ to n **do**
  for each memory item $m \in M_j$ **do**
    $t \leftarrow \phi(m)$
    $N_t \mathrel{+}= \frac{s_j}{|M_j| \cdot n}$
  end for
end for
for $t = 1$ to T **do**
  $r_{EARA}^{(t)} \leftarrow (1-\beta)\frac{r_{global}}{T} + \beta\, N_t$
end for
return $\{r_{EARA}\}$

ing coherent integration across the entire sequence.

3.2 Evidence-Anchored Reward Attribution

Although dense chunk-level step rewards capture the local quality of individual memory operations and alleviate reward sparsity, they remain insufficient to reflect the long-term utility of memory management across an entire chunk stream. To reasonably attribute the global QA signal to the memory operations that truly matter, we introduce **Evidence-Anchored Reward Attribution (EARA)**, a mechanism that bridges the final global reward (r_{global}) and intermediate memory actions by tracing downstream reasoning back to the specific memory operations that enabled them. EARA is instantiated through two complementary mechanisms, which is summarized in Algorithm 1.

We first quantify the objective utility of each memory operation step based on its evidential support for downstream reasoning. Consider a rollout consisting of T memory steps and evaluated by n global QA queries. Let $s_j \in [0, 1]$ denote the score obtained for question q_j, and let M_j be the set of memory entries retrieved by the reasoning agent to produce its answer. We define the *Normalized Evidence Contribution (NEC)* of step t as:

$$N_t = \sum_{j=1}^{n} \sum_{\substack{m \in M_j \\ \phi(m)=t}} \frac{s_j}{|M_j| \cdot n}, \tag{3}$$

where $\phi(m)$ identifies the memory operation step from which memory item m originates.

To prevent over-sensitivity to individual evaluation questions and to further stabilize gradient estimation, we adopt a soft reward redistribution strategy. Specifically, the final EARA reward assigned to step t is computed as a combination of a uniform baseline reward and the evidence-driven contribution:

$$r_{EARA}^{(t)} = (1-\beta)\frac{r_{global}}{T} + \beta N_t, \tag{4}$$

where $\beta \in [0, 1]$ is an attribution factor controlling the strength of evidence-based credit assignment. The uniform term provides a *participation credit* to all steps in the rollouts, ensuring a stable learning signal for exploration, while the NEC term assigns *performance-based credit* to the specific memory operations that produced effective evidence. Importantly, EARA redistributes the full global reward of each rollout across all memory steps, as shown in Appendix B.2, ensuring the attribution remains consistent with the original optimization objective.

3.3 Policy Optimization via GRPO

Beyond correctness from r_{chunk} and r_{EARA}, manager operations are also encouraged to be efficient in terms of memory usage and comply with the formatting required for deployment. To this end, we incorporate auxiliary rewad for invalid function formatting (r_{fmt}) and memory compression (r_{comp}), which have been shown effective in prior work (Wang et al., 2025). The total reward r_t for a set of operations P_t is formulated as a weighted sum:

$$r_t = r_{EARA}^{(t)} + r_{fmt}^{(t)} + w_1 r_{chunk}^{(t)} + w_2 r_{comp} \tag{5}$$

where w_1 and w_2 are balancing hyperparameters for the auxiliary rewards, each normalized to lie in the range $[0, 1]$. Formal definitions of all reward components are provided in Appendix B.3.

We then optimize the policy using GRPO (Shao et al., 2024). At each step t, the simplified objective with memory operations is:

$$\mathcal{L}_{\text{policy}} \approx \sum_{G} \log \rho_\theta(P_t \mid \mathcal{M}_{t-1}, c_t) \cdot A_t, \tag{6}$$

where G represents the number of rollouts, P_t denotes the operations, A_t is the corresponding advantage and $\rho_\theta(\cdot)$ is the corresponding importance ratio. The complete GRPO objective is introduced in Appendix C.

Method	AR			TTL			LRU	Avg. ↑	Len. ↓
	SQuAD	HotpotQA	PerLTQA	TREC-C	NLU	Pubmed	BookSum		
Non-Constructive Memory									
Long-Context	0.742	<u>0.852</u>	0.605	0.623	<u>0.708</u>	0.533	0.052	0.588	10.8K
RAG-Top2	0.762	0.849	0.623	0.612	0.508	<u>0.570</u>	0.042	0.567	11.3K
Workflow-Based Memory Systems									
A-Mem	<u>0.827</u>	0.826	0.227	0.628	0.686	0.443	0.181	0.545	11.5K
LightMem	0.214	0.271	0.283	**0.833**	0.671	0.214	0.154	0.377	2.12K
Train-Based Memory Agents									
MemAgent	0.091	0.140	0.052	0.562	0.290	0.343	0.103	0.226	<u>0.84K</u>
MEM1	0.039	0.083	0.068	0.269	0.056	0.175	0.085	0.111	**0.17K**
Mem-α	0.786	0.832	<u>0.659</u>	0.666	0.658	0.545	<u>0.187</u>	<u>0.619</u>	7.90K
Fine-Mem	**0.836**	**0.872**	**0.693**	<u>0.675</u>	**0.728**	**0.624**	**0.213**	**0.663**	10.2K

Table 1: Performance in the memory on validation datasets of Memalpha. **Bold** and <u>underline</u> numbers indicate the best and second performance among evaluated methods. The **Len.** denotes the average memory length per sample in thousands of tokens.

4 Experimental Setup

4.1 Datesets and Evaluation

Datasets Following Mem-α (Wang et al., 2025), we train our model on the Memalpha training corpus, specifically augmented with constructed chunk-level QA pairs. We conduct a comprehensive evaluation to assess both in-distribution (ID) performance, using the Memalpha validation set, and out-of-distribution (OOD) generalization, using MemoryAgentBench (Hu et al., 2025a), which features significantly longer contexts and higher complexity compared to existing benchmarks like LoCoMo (Maharana et al., 2024) and LongMemEval (Wu et al., 2024a).

Metrics We assess three core capabilities of memory mechanisms utilizing the corresponding sub-datasets from MemoryAgentBench: (1) **Accurate Retrieval (AR)**: This category includes Single-Doc and Multi-Doc measured by *Substring Exact Match*, and LME(S*) measured by *LLM-as-Judge*. (2) **Test-Time Learning (TTL)**: Covers five classification domains (TREC-C, NLU, TREC-F, Clinic, Banking77), evaluated via *Classification Accuracy*. (3) **Long-Range Understanding (LRU)**: Utilizes InfBench-Sum for summarization tasks. Detailed descriptions and statistics for all datasets are provided in Appendix D.

4.2 Baselines

To validate the effectiveness of Fine-Mem, we compare it against seven baselines, which can be grouped into three major categories, with detailed settings provided in Appendix E.1:

- **Non-Constructive Memory**. These methods operate without explicit memory construction and rely solely on raw context or simple retrieval mechanisms. (1) **Long-Context** strategy directly processes the entire memory context using its maximum window size.

  (2) **RAG-Top2** baseline applies a retrieval-augmented strategy based on BM25 to select the two most relevant chunks, which are then used to answer the query.

- **Workflow-Based Memory Systems**. This category builds external memory modules through LLM-driven workflows. (1) **A-Mem** (Xu et al., 2025) is a dynamic agentic memory system that creates, links, and updates structured memories to support cross-session reasoning. (2) **LightMem** (Fang et al., 2025) is a cognitively-inspired system that optimizes efficiency through compressive sensory filtering, topic-aware consolidation, and decoupled sleep-time update.

- **Train-Based Memory Agents**. These approaches formulate memory management as a learnable decision-making process. (1) **MemAgent** (Yu et al., 2025) is trained to iteratively process all available chunks according to a task description and forms an internal memory state from which answers are produced. (2) **MEM1** (Zhou et al., 2025) functions as an agent that maintains a single paragraph of memory, which it continuously retrieves and updates as new information becomes available. (3) **Mem-α** (Wang et al., 2025) employs a GRPO-based training objective to optimize an external memory manager

Method	AR			TTL					LRU	Avg. ↑	Len. ↓
	Single-Doc	Multi-Doc	LME(S)	TREC-C	NLU	TREC-F	Clinic	Banking77	InfBench		
Non-Constructive Memory											
Long-Context	0.280	0.270	0.292	0.640	0.740	0.340	0.860	0.770	0.125	0.480	33K
RAG-Top2	0.690	0.450	**0.581**	0.690	0.650	0.210	0.700	0.750	0.065	0.532	209K
Workflow-Based Memory Systems											
A-Mem	0.420	0.360	0.427	0.680	0.720	0.400	0.170	0.780	0.103	0.451	210K
LightMem	0.270	0.320	0.420	0.450	0.610	0.250	0.490	0.480	0.010	0.367	42.5K
Train-Based Memory Agents											
MemAgent	0.070	0.160	0.050	0.370	0.260	0.210	0.250	0.370	0.043	0.198	0.92K
MEM1	0.070	0.180	0.090	0.180	0.000	0.000	0.090	0.000	0.029	0.071	**0.21K**
Mem-α	0.740	0.680	0.520	**0.710**	0.710	0.410	0.730	0.700	0.129	0.592	129K
Fine-Mem	**0.760**	**0.720**	0.550	0.690	**0.800**	**0.590**	**0.890**	**0.820**	**0.153**	**0.664**	148K

Table 2: Performance in the memory on MemoryAgentBench. **Bold** and underline numbers indicate the best and second performance among evaluated methods. The **Len.** denotes the average memory length per sample in thousands of tokens.

operating over a hierarchical memory system.

4.3 Implementation Details

We implement Fine-Mem by building upon the VERL framework (Sheng et al., 2024). During main experiments, we adopt BM25 for retrieval, utilize `Qwen3-4B` as the backbone manager model, and deploy a long-context `Qwen3-32B` model via vLLM (Kwon et al., 2023) as the reasoning agent. We set the learning rate to 1×10^{-6}, the batch size to 32, and grpo_rollout_n to 8, following the settings used in Mem-α. The models are trained for 2 epochs, and we report the performance of the final checkpoint. The reward weights in Eq.5 are configured as $w_1 = 0.5, w_2 = 0.05, \beta = 0.5$. Further implementation details of Fine-Mem, please refer to Appendix E.2.

5 Results

5.1 Main Results

We report the main experimental results in Tables 1 and 2. Fine-Mem achieves superior overall performance, scoring the highest averages of 0.663 on MemAlpha and 0.664 on MemoryAgentBench. It consistently attains either the best or second-best results across all sub-capabilities, including Accurate Retrieval, Test-Time Learning, and Long-Range Understanding.

Compared to the state-of-the-art baseline Mem-α, Fine-Mem demonstrates consistent improvements, achieving an average improvement of 4.4% on Memalpha and 7.2% on MemoryAgentBench, while maintaining a compact memory length relative to the total input chunks. These results validate the effectiveness of incorporating fine-grained supervision into the *memory manager*, suggesting that precise training signals are essential for learning memory management.

Method	Metric	Memalpha	MAB.	Avg.
OR-Based	Perf. ↑	0.648	0.605	0.627
	Len. ↓	10.7K	174K	92.4K
w/ *CSR*	Perf. ↑	0.657	0.621	0.639
	Len. ↓	13.4K	159K	86.2K
w/ *EARA*	Perf. ↑	0.641	0.603	0.622
	Len. ↓	**9.31K**	**112K**	**60.7K**
Fine-Mem	Perf. ↑	**0.663**	**0.664**	**0.663**
	Len. ↓	10.2K	148K	79.1K

Table 3: Ablation study of different components in Fine-Mem on Memalpha and MemoryAgentBench (**MAB.**) datasets. **Bold** and underlined numbers indicate the best and second results for each metric.

In contrast, workflow-based memory agents, despite utilizing strong LLM backbones, suffer from rigid behavioral patterns that limit their generalization to complex tasks. For instance, while LightMem maintains a substantial memory length of 42.5K tokens on MemoryAgentBench, its reliance on iterative pre-compression for long chunks leads to information loss, degrading its performance on the Accurate Retrieval subset. Meanwhile, train-based methods like Mem1 and MemoryAgent are designed to efficiently fold information of working memory but struggle to adequately scale to external long-term memory management.

Overall, these results demonstrate that *Fine-Mem consistently delivers robust performance for long-term memory management*.

5.2 Ablation Studies

In this section, we present ablation studies to evaluate the contribution of each Fine-Mem module and

w_1	Memalpha				MemoryAgentBench				Avg.	
	AR	TTL	LRU	Len.	AR	TTL	LRU	Len.	Perf. ↑	Len. ↓
0.0	0.767	0.662	0.202	9.31K	0.463	**0.776**	**0.160**	**112K**	0.505	**60.7K**
0.5	**0.800**	0.676	0.213	10.2K	**0.677**	0.758	0.153	148K	**0.546**	79.1K
1.0	0.773	0.649	0.180	17.4K	0.478	0.760	0.141	150K	0.497	83.7K

Table 4: Ablation study on the hyperparameters w_1 in total reward on Memalpha and MemoryAgentBench, while keeping w_2 fixed at 0.05.

w_2	Memalpha				MemoryAgentBench				Avg.	
	AR	TTL	LRU	Len.	AR	TTL	LRU	Len.	Perf. ↑	Len. ↓
0.05	**0.800**	0.676	0.213	10.2K	**0.677**	0.758	0.153	148K	**0.546**	79.1K
0.10	0.778	**0.680**	**0.219**	8.20K	0.620	0.742	0.141	165K	0.530	86.6K
0.20	0.781	0.661	0.195	**7.91K**	0.581	0.756	0.138	151K	0.519	79.5K

Table 5: Ablation study on the hyperparameters w_2 in total reward on Memalpha and MemoryAgentBench, while keeping w_1 fixed at 0.5.

the impact of hyperparameters related to the total reward and EARA.

Ablation of Fine-Mem Modules. We evaluate three variants of Fine-Mem to isolate the contributions of specific modules: 'OR-Based' (outcome reward only), 'w/ CSR' (adding Chunk-level Step Reward), and 'w/ EARA' (adding Evidence-Anchored Reward Attribution). As detailed in Table 3, incorporating CSR improves the manager's average performance from 0.627 to 0.639. However, this gain comes at the cost of a relatively larger memory length. Conversely, the EARA mechanism proves highly effective for compression, achieving the lowest average memory length of 60.7K. Yet, when used in isolation, its sparse reward signals lead to a performance drop to 0.622. By combining both modules, Fine-Mem achieves the best overall performance of 0.663 while maintaining a controlled average memory size of 79.1K. This result confirms that the two mechanisms are mutually complementary, effectively balancing high retrieval accuracy with memory efficiency.

Ablation of Total Reward Hyperparameters. We investigate the influence of the total reward weights w_1 and w_2 on the balance between dense process rewards and compression-based rewards, as outlined in Eq. 5 and summarized in Table 4 and 5. Our findings indicate that a high value of w_1 encourages excessive per-step memorization, leading to longer memory lengths, which negatively affects the performance of the memory manager.

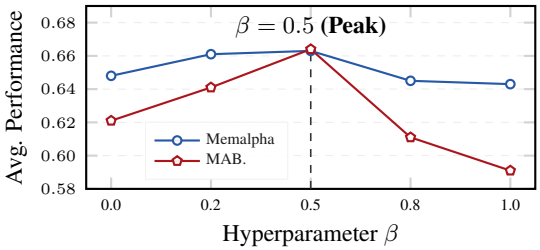

Figure 3: Ablation study on the hyperparameter β in Evidence-Anchored Reward Attribution on Memalpha and MemoryAgentBench (**MAB.**)

Conversely, increasing w_2 enhances memory compactness but risks discarding crucial information. This leads to performance degradation and, notably, results in longer memory lengths on the out-of-distribution dataset MemoryAgentBench. Based on these observations, we set $w_1 = 0.5$ and $w_2 = 0.05$ as the default configuration, which provides an optimal trade-off between memory length and downstream task performance.

Ablation of the Factor in EARA. We conduct a comprehensive sensitivity analysis of EARA with respect to the hyperparameter β, as visualized in Figure 3. We observe that the performance peaks at $\beta = 0.5$, indicating that a balanced reward assignment is essential. Specifically, increasing β beyond 0.5 leads to significant performance degradation, particularly on the out-of-distribution MemoryAgentBench. A high β may allocate the most global reward to specific steps, which induces supervisory sparsity and hinders generalization to

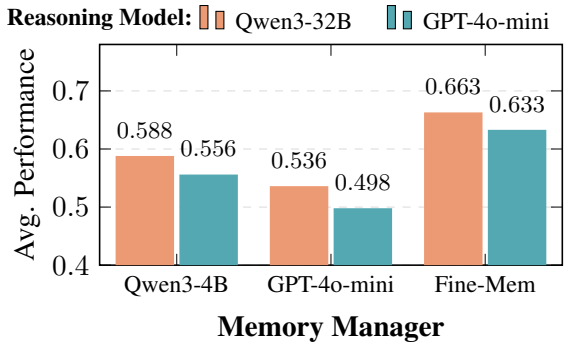

Figure 4: Performance comparison of different Memory Managers combined with varying Reasoning Models on the Memalpha dataset.

Model	Methods	Memalpha	MAB.	Avg. ↑	Len. ↓
Qwen3-4B	Base	0.588	0.615	0.601	61.4K
	Fine-Mem	0.663	0.664	0.663 ↑	79.1K
Qwen3-1.7B	Base	0.482	0.500	0.491	42.5K
	Fine-Mem	0.623	0.626	0.625 ↑	73.7K
Llama3.2-3B	Base	0.630	0.643	0.637	69.6K
	Fine-Mem	0.649	0.657	0.653 ↑	82.1K

Table 6: Performance of different models with Fine-Mem on Memalpha and MemoryAgentBench (**MAB.**).

out-of-distribution tasks. Conversely, a low β setting ignores the contribution of key memory operations, providing insufficient guidance to the manager and thereby hindering the acquisition of effective memory maintenance strategies. Consequently, we adopt $\beta = 0.5$ as the default configuration across all main results and ablation studies.

5.3 Further Analysis

Generalization of the Memory Manager. To evaluate the generalization of memory managers, we benchmark Fine-Mem against baselines (`Qwen3-4B` and `GPT-4o-mini`) across different reasoning models, as illustrated in Figure 4 and Figure 5. The results demonstrate that Fine-Mem consistently achieves the highest average performance regardless of the reasoning model employed. Specifically, it reaches a score of 0.663 with `Qwen3-32B`, significantly surpassing the baselines, and maintains a similar advantage with `GPT-4o-mini`. This confirms the robustness and effectiveness of Fine-Mem across diverse downstream architectures.

Impact on Different Backbones. Furthermore, we assess the effectiveness of Fine-Mem by applying it to specific backbone models, including `Qwen3-4B`, `Qwen3-1.7B`, and `Llama3.2-3B`. As shown in Table 6, Fine-Mem yields consistent average performance improvements across these architectures. These results further confirm the robustness and general applicability of Fine-Mem, indicating its potential to enhance models of varying scales and types.

6 Related Work

Memory-Augmented LLM Agents. Effective memory mechanisms are crucial for LLM agents to maintain coherent reasoning in long-horizon scenarios (Zhang et al., 2025a; Wang et al., 2024; Du et al., 2025). Early approaches (Modarressi et al., 2023; Wang et al., 2023), including MemGPT (Packer et al., 2023) and MemoryBank (Zhong et al., 2024), adopted retrieval-augmented paradigms that offloaded interaction history to external memory. Building on this foundation, subsequent work introduced structural optimizations. Frameworks like MemTree (Rezazadeh et al., 2024) and Zep (Rasmussen et al., 2025) organize context into hierarchical or temporal structures, while graph-based architectures (Chhikara et al., 2025; Gutiérrez et al., 2025) explicitly model relational dependencies between memory nodes. More recently, methods like MIRIX (Wang and Chen, 2025), and A-Mem (Xu et al., 2025) have shifted toward autonomous update schemes, where agents actively compress and prune information to optimize the retention-efficiency trade-off (Fang et al., 2025), rather than passively accumulating history. However, these methods depend on rigid workflows with strong LLMs, resulting in high overhead and low scalability (Hu et al., 2025a).

Reinforcement Learning for Memory Agent. To enhance the adaptability of memory systems, recent research has increasingly integrated RL with LLMs (Zhang et al., 2023, 2025b; Hu et al., 2025b), reframing memory management as a learnable policy rather than a static, rule-based mechanism. One prominent research direction focuses on training agents to autonomously manage their intrinsic working context. Approaches such as MEM1 (Zhou et al., 2025) and MemAgent (Yu et al., 2025) leverage RL to actively compress and reorganize information, thereby empowering agents to effectively navigate long-horizon dependencies. Conversely, a parallel line of inquiry adopts a decoupled architecture by introducing a dedicated memory controller. Systems like Memory-R1 (Yan et al., 2025) and Mem-α (Wang et al., 2025) employ a specialized manager to direct storage and retrieval

operations independently of the reasoning agent. Despite advancements, robust memory agents remain challenging due to sparse rewards and inefficient credit assignment in long-horizon tasks.

7 Conclusion

This paper presents Fine-Mem, a unified reinforcement learning framework established for long-horizon memory management. Through the integration of the newly proposed Chunk-level Step Reward and Evidence-Anchored Reward Attribution, the approach effectively mitigates the challenges of reward sparsity and delayed credit assignment. Empirical evaluations on comprehensive benchmarks indicate that Fine-Mem consistently outperforms strong baselines. Further ablation studies and hyperparameter analyses validate the efficacy and stability of Fine-Mem. Finally, experiments across diverse reasoning models and manager backbones validate the framework's strong generalization capabilities in various settings.

Limitations

This work exhibits several limitations worth noting. **Firstly**, our retrieval mechanism relies on BM25, despite being a simple and effective baseline for long-term memory, which limits the capture of deeper semantic relationships. Future research will explore richer alternatives, such as dense vector or graph-based retrieval. **Secondly**, this work exclusively focuses on memory management within the textual mode. We aim to extend our approach to support multimodal memory in future work, thereby broadening its real-world utility. **Thirdly**, although the decoupled training strategy effectively optimizes the memory manager, it does not support the enhancement of the reasoning model. Future research could investigate a co-evolutionary paradigm to achieve the synergistic optimization of both the manager and the reasoning agent.

Acknowledgments

Xiaocheng Feng, Jun Xu, Jiuchong Gao and Bing Qin are the co-corresponding authors of this work. We thank the anonymous reviewers for their insightful comments. This work was supported by the National Natural Science Foundation of China (NSFC) (grant 62522603, 62276078), the Key R&D Program of Heilongjiang via grant 2022ZX01A32, the Fundamental Research Funds for the Central Universities (XNJKKGYDJ2024013) .

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. *arXiv preprint arXiv:2003.04807*.

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, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z Pan. 2025. Rethinking memory in ai: Taxonomy, operations, topics, and future directions. *arXiv preprint arXiv:2505.00675*.

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.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. 2025. Lightmem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*.

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. 2025a. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. 2025b. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. Booksum: A collection of datasets for long-form narrative summarization. In *Findings of the association for computational linguistics: EMNLP 2022*, pages 6536–6558.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Jiafeng Liang, Hao Li, Chang Li, Jiaqi Zhou, Shixin Jiang, Zekun Wang, Changkai Ji, Zhihao Zhu, Runxuan Liu, Tao Ren, Jinlan Fu, See-Kiong Ng, Xia Liang, Ming Liu, and Bing Qin. 2025. Ai meets brain: Memory systems from cognitive neuroscience to autonomous agents. *Preprint*, arXiv:2512.23343.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. 2025. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2021. Benchmarking natural language understanding services for building conversational agents. In *Increasing naturalness and flexibility in spoken dialogue interaction: 10th international workshop on spoken dialogue systems*, pages 165–183. Springer.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. 2025. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*.

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*.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. 2023. Ret-llm: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*.

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*.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. 2024. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for llms. *arXiv preprint arXiv:2410.14052*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. 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. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2023. Enhancing large language model with self-controlled memory framework. *arXiv preprint arXiv:2304.13343*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

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

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

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

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

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, et al. 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, et al. 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, et al. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent, 2025. *URL https://arxiv. org/abs/2507*, 2259.

Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. 2023. Large language models are semi-parametric reinforcement learning agents. *Advances in Neural Information Processing Systems*, 36:78227–78239.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. 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, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025a. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47.

Zeyu Zhang, Quanyu Dai, Rui Li, Xiaohe Bo, Xu Chen, and Zhenhua Dong. 2025b. Learn to memorize: Optimizing llm-based agents with adaptive memory framework. *arXiv preprint arXiv:2508.16629*.

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 Details of Memory Operations

Consistent with prior work (Wang et al., 2025; Yan et al., 2025), we define the operation space of the memory manager as INSERT, UPDATE, DELETE, SKIP. To align the manager's behavior with these operations, we design a system prompt that constrains the model to produce outputs in a JSON-based function-call format, as illustrated in Figure 9. When the SKIP operation is selected, the model is instructed to return the token "done". For the INSERT, UPDATE, and DELETE operations, the prompt enforces structured JSON schemas that explicitly specify the operation type, target object, and corresponding content, as detailed in Figures 6, 7, 8, respectively.

B Details of Reward Function

B.1 Chunk-Level QA Construction

To generate high-quality, verifiable supervision signals for RL, we design a multi-stage data engineering pipeline, as summarized in Algorithm 2.

For each chunk in an instance, we first prompt `GPT-4o-mini` to extract key information and convert it into QA pairs. The prompt is carefully crafted to encourage the generation of concise, factoid-style answers (e.g., entities, dates), which substantially reduces ambiguity in automated evaluation compared to open-ended generation. Moreover, to promote the reasoning agent's ability to generalize in downstream tasks based on the memory, we allow paraphrases of the content within the QA pairs, as detailed in Figures 10.

Crucially, we incorporate a model-in-the-loop verification step to ensure data quality. Specifically, we employ `Qwen3-32B` as a verifier, asking it to answer each generated question using only the content of the corresponding chunk. A QA pair is considered valid only if the verifier can correctly derive the answer from the source text. This procedure ensures that the resulting dense rewards promote the extraction of information genuinely accessible from the text.

After removing duplicates with historical questions, we maintain a fixed budget of five QA pairs per chunk. This offline pre-computation avoids the latency of real-time LLM evaluation, supporting efficient and scalable training.

Algorithm 2 Construction of Chunk-level QA Dataset

Input: Dataset $\mathcal{D}$, Teacher π_{tea} (GPT-4o-mini), Verifier π_{ver} (Qwen3-32B)
Input: Hyperparameter: Top-K ($K = 5$)
Output: Augmented Dataset with Chunk-level QA pairs $\mathcal{D}_{\text{chunk}}$

Initialize $\mathcal{D}_{\text{chunk}} \leftarrow \emptyset$
for each instance $I = \{c_1, c_2, \ldots, c_T\}$ in $\mathcal{D}$ **do**
  $H_{\text{qa}} \leftarrow \emptyset$; $I_{\text{chunk}} \leftarrow \emptyset$
  for each chunk c_t in I **do**
    $\mathcal{Q}_{\text{raw}} \leftarrow \pi_{\text{tea}}(c_t)$
    $\mathcal{Q}_{\text{verified}} \leftarrow \emptyset$
    for (q, a) in $\mathcal{Q}_{\text{raw}}$ **do**
      $\hat{a} \leftarrow \pi_{\text{ver}}(q, c_t)$
      if Match$(\hat{a}, a)$ **and** $(q, a) \notin H_{\text{qa}}$ **then**
        $\mathcal{Q}_{\text{verified}}$.add$((q, a))$
        H_{qa}.add$((q, a))$
      end if
    end for
    $\mathcal{G}_t \leftarrow$ SelectTopK$(\mathcal{Q}_{\text{verified}}, K)$
    I_{chunk}.append$((c_t, \mathcal{G}_t))$
  end for
  $\mathcal{D}_{\text{chunk}}$.add$(I_{\text{chunk}})$
end for
return $\mathcal{D}_{\text{chunk}}$

B.2 Proof of EARA Reward

Consider a single rollout consisting of T memory update steps and n global QA pairs. For the j-th question, let M_j denote the set of retrieved memory items, and let $\phi(m)$ map a memory item m to the update step at which it was generated.

Let s_j be the score associated with the j-th question. We define the *Normalized Evidence Contribution (NEC)* for step t as:

$$N_t = \sum_{j=1}^{n} \sum_{\substack{m \in M_j \\ \phi(m)=t}} \frac{s_j}{|M_j| \cdot n}. \tag{7}$$

This quantity aggregates the evidence-based credit assigned to step t from all questions, where each question's score is evenly distributed over its retrieved memory items and normalized by the total number of questions.

Summing N_t over all update steps yields:

$$\begin{aligned}\sum_{t=1}^{T} N_t &= \sum_{t=1}^{T}\sum_{j=1}^{n}\sum_{\substack{m\in M_j\\ \phi(m)=t}} \frac{s_j}{|M_j|\cdot n} \\ &= \sum_{j=1}^{n}\frac{s_j}{n}\sum_{t=1}^{T}\sum_{\substack{m\in M_j\\ \phi(m)=t}} \frac{1}{|M_j|} \\ &= \sum_{j=1}^{n}\frac{s_j}{n}\sum_{m\in M_j}\frac{1}{|M_j|} \\ &= \frac{1}{n}\sum_{j=1}^{n} s_j.\end{aligned} \tag{8}$$

Since $\sum_{m\in M_j}\frac{1}{|M_j|} = 1$ for j-th question, and the rollout-level reward is defined as $r_{global} = \frac{1}{n}\sum_{j=1}^{n} s_j$, it follows that:

$$\sum_{t=1}^{T} N_t = r_{global}. \tag{9}$$

EARA assigns the following reward to each update step t:

$$r_{EARA}^{(t)} = (1-\beta)\frac{r_{global}}{T} + \beta N_t, \tag{10}$$

where $\beta \in [0,1]$ controls the trade-off between uniform and evidence-based credit assignment.

Summing the step-level rewards over the entire rollout gives:

$$\begin{aligned}\sum_{t=1}^{T} r_{EARA}^{(t)} &= (1-\beta) r_{global}\sum_{t=1}^{T}\frac{1}{T} + \beta\sum_{t=1}^{T} N_t \\ &= (1-\beta) r_{global} + \beta r_{global} \\ &= r_{global}.\end{aligned} \tag{11}$$

Therefore, EARA achieves exact reward conservation, redistributing the rollout-level reward to step-level rewards without altering the total training signal. □

B.3 Hybrid Reward

In this section, we introduce the mathematical formulations of the four components constituting our hybrid reward function: EARA for global QA Reward (r_{EARA}), Chunk-Level Step Reward (r_{chunk}), Formatting Validity Reward (r_{fmt}), and Compression Efficiency Reward (r_{comp}), referring to Mem-α. We define the memory state at time step t as $\mathcal{M}_t$, and the manager's operations applied to the incoming chunk c_t as P_t.

EARA for Global QA Reward This reward measures the long-term consistency of the memory system with respect to downstream task performance. Given a global validation set $\mathcal{Q}\text{glob} = \{(q_i, y_i)\}_{i=1}^{N}$ derived from downstream tasks that require long-range reasoning, the reward is evaluated only at the final memory state $\mathcal{M}_T$. For each query q_i, the reasoning agent π_{reason} generates an answer conditioned on the final memory $\mathcal{M}_T$, which is denoted as $\pi_{\text{reason}}(\cdot \mid \mathcal{M}_T, q_i)$. The global QA reward is defined as:

$$r_{global} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left[\pi_{\text{reason}}(\cdot \mid \mathcal{M}_T, q_i), y_i\right] \tag{12}$$

where $\mathbb{I}[\cdot]$ denotes an indicator function that returns 1 if the prediction produced by the reasoning agent π_{reason} is judged correct under the evaluation metric of the corresponding downstream task, and 0 otherwise. To enable fine-grained learning, we adopt *Evidence-Anchored Reward Attribution (EARA)*, which redistributes the global reward across individual memory construction steps ($r_{EARA}^{(t)}$) according to their contribution to successful reasoning outcomes, as shown in section 3.2.

Chunk-level Step Reward To mitigate the supervision sparsity inherent in long-horizon tasks, we introduce a chunk-level step reward focused on local information retention. For each incoming chunk c_t, we automatically construct a set of local QA pairs $\mathcal{Q}_{local}^{(t)} = \{(q_j^{(t)}, y_j^{(t)})\}_{j=1}^{K}$ derived solely from the content of c_t. The reward at step t verifies whether the updated memory $\mathcal{M}_t$ effectively retains key information based on the reasoning agent π_{reason}:

$$r_{chunk}^{(t)} = \frac{1}{K}\sum_{j=1}^{K}\mathbb{I}\left[\pi_{\text{reason}}(\cdot|\mathcal{M}_t, q_j^{(t)}), y_j^{(t)}\right] \tag{13}$$

This component incentivizes the agent to maximize immediate information precision, penalizing operations that discard essential details before global evaluation is possible.

Formatting Validity Reward To guarantee the operational integrity of the system during deployment, we enforce strict adherence to the defined schema. Let $\mathcal{P}_{valid}$ denote the set of permissible operation definitions and syntactic rules. For a sequence of operations $P_t = \{p_{t,1}, p_{t,2}, \ldots, p_{t,M}\}$ generated at step t, the formatting reward is calculated as the average validity of the individual

operations:

$$r_{fmt}^{(t)} = \frac{1}{M}\sum_{i=1}^{M} v(p_{t,i}),$$
$$\text{where } v(p_{t,i}) = \begin{cases} 1 & \text{if } p_{t,i} \in \mathcal{P}_{valid} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

This constraint ensures the policy learns to generate structurally sound memory updates that align with the environment's interface requirements.

Compression Efficiency Reward This reward regulates the trade-off between retention and storage, preventing memory size from scaling linearly with the input stream. Let $L(\cdot)$ denote the token length function. We define compression efficiency by comparing the final memory against the cumulative input size:

$$r_{comp} = 1 - \frac{L(\mathcal{M}_T)}{\sum_{t=1}^{T} L(c_t)} \tag{15}$$

This objective drives the policy towards succinct representations (e.g., abstraction or summarization) rather than copying.

Consequently, the total hybird reward r_t for the sequence of operations P_t is defined as a weighted linear combination:

$$r_t = r_{EARA}^{(t)} + r_{fmt}^{(t)} + w_1 r_{chunk}^{(t)} + w_2 r_{comp} \tag{16}$$

where w_1 and w_2 denote hyperparameters that balance the contributions of these distinct reward components.

C Detailed GRPO Formulation

In this section, we provide the complete formulation of the GRPO objective used to train the *Memory Manager*.

For each input chunk c_t at step t, we sample a group of G operations $\{P_{t,1}, \ldots, P_{t,G}\}$ from the old policy π_{old}. For each sampled sequence $P_{t,j}$, we compute a hybrid reward $r_{t,j}$, which evaluates the overall quality of the operations performed at step t. The group-relative advantage for the j-th sequence is then obtained by standardizing these rewards:

$$A_{t,j} = \frac{r_{t,j} - \mu_{\text{group}}^{(t)}}{\sigma_{\text{group}}^{(t)} + \epsilon}, \tag{17}$$

where $\mu_{\text{group}}^{(t)}$ and $\sigma_{\text{group}}^{(t)}$ denote the mean and standard deviation of rewards across the group of operation sequences at step t, respectively.

The final GRPO objective is defined as:

$$\mathcal{J}(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{t=1}^{G}\frac{1}{|P_t|}\sum_{i=1}^{|P_t|}\min\Big(\rho_t(\theta),\ \text{clip}\big(\rho_t(\theta), 1-\varepsilon, 1+\varepsilon\big)\Big)A_t\Bigg],$$
$$\text{where}\quad \rho_t(\theta) = \frac{\pi_\theta(P_t \mid \mathcal{M}_{t-1}, c_t)}{\pi_{\text{old}}(P_t \mid \mathcal{M}_{t-1}, c_t)},$$
$$P_t = \{P_{t,1}, \ldots, P_{t,G}\}. \tag{18}$$

Following standard practices, we employ the clipping to constrain the policy update and integrate a KL-divergence term into the loss function.

D Details of Datasets

In this section, we provide an overview of the training dataset and evaluation datasets. Following Mem-α (Wang et al., 2025), we foucs on memory agents through three core competencies: **Accurate Retrieval** (AR), which measures the agent's ability to store factual information and retrieve it precisely from memory faithfully; **Test-Time Learning** (TTL), which assesses the capacity to acquire and apply new rules or patterns introduced dynamically during interaction; **Long-Range Understanding** (LRU), which evaluates the ability to maintain global context over long horizons and synthesize a coherent understanding.

D.1 Training Dataset

We construct a training corpus based on the data used in Mem-α, further augmented with a newly constructed chunk-level QA dataset. The combined datasets form a multi-task mixture aligned with three core memory-related competencies: Accurate Retrieval (AR), Test-Time Learning (TTL), and Long-Range Understanding (LRU). (1) The AR category includes SQuAD (Rajpurkar et al., 2016), HotpotQA (Yang et al., 2018), PerLTQA (Du et al., 2024), and LME-Train (Wu et al., 2024b); (2) The TTL category adopts classification-oriented datasets such as PubMed-RCT (Dernoncourt and Lee, 2017), NLU, and TREC-Coarse (Hu et al., 2025a); (3) The LRU category employs the BookSum dataset (Kryściński et al., 2022), segmented into sequential chunks. All datasets are presented in a sequential format, where information is organized incrementally across chunks to facilitate memory accumulation. The resulting statistics are summarized in Table 7.

Category	Dataset	Ins.	ChunkQA
AR	SQuAD	100	4975
	HotpotQA	100	4840
	PerLTQA	27	3140
	LME-Train	50	3845
TTL	NLU	49	2450
	TREC-Coarse	51	2550
	PubMed-RCT	90	4500
LRU	BookSum	100	3915
Total		**567**	**30215**

Table 7: Statistics of the training dataset. **AR**: Accurate Retrieval, **TTL**: Test-Time Learning, **LRU**: Long-Range Understanding. **Ins.** denotes the number of instances, and **ChunkQA** denotes the number of chunk-level QA .

D.2 Evaluation Dataset

Aligned with the Mem-α, our evaluation assesses both in-distribution (ID) and out-of-distribution (OOD) performance. The ID evaluation utilizes the Mem-α validation set, comprising 468 instances across 7 subsets (excluding LME-Train used in training); The OOD evaluation employs MemoryAgentBench (Hu et al., 2025a), which features 112 instances across 9 datasets characterized by significantly longer context lengths. Specifically for MemoryAgentBench, the datasets are aligned with three core competencies: (1) The AR category includes RULER-QA1 and RULER-QA2 (Hsieh et al., 2024), which test single-document and multi-document question-answering capabilities, respectively, as well as LME(S*) (Wu et al., 2024a) for evaluating precise information retrieval; (2) The TTL category adopts five classification datasets covering varying granularities: TREC-Coarse and TREC-Fine (Li and Roth, 2002), NLU (Liu et al., 2021), CLINIC150 (Larson et al., 2019), and Banking77 (Casanueva et al., 2020); (3) The LRU category uses the InfBench-Sum dataset (Zhang et al., 2024), which requires high-level summarization from full-length novels.

We assess the performance across memory capabilities using five complementary metrics according to the nature of each task: (1) **SubEM (Subsection Exact Match)**: Measures strict retrieval recall by checking if the ground-truth answer appears verbatim in the responses. (2) **EM (Exact Match)**: Evaluates classification or rigid-format tasks by computing the percentage of exact matches between predicted and ground-truth outputs. (3) **Source-based**: Verifies whether responses are correctly derived from specific memory chunks, ensuring factual support from the source rather than hallucination. (4) **LLM Judge (LLM-J)**: Uses Qwen3-32B to semantically assess open-ended outputs against references, focusing on meaning rather than exact string matches. (5) **KW Hit (Keyword Hit)**: Measures the recall of key entities, events, or concepts from the reference in the generated output, reflecting information coverage and synthesis.

The statistics for all evaluation datasets are summarized in Table 8.

E Details of Implementation

E.1 Baselines

In this section, we describe the detailed implementation of all baseline methods used in our evaluation, which can be grouped into three categories, comprising seven approaches. Unless otherwise specified, we adopt **Qwen3-32B** with a **32k-token context window** as the reasoning agent for all baselines and uniformly deploy it in a **non-thinking** inference mode.

- **Non-Constructive Memory**.

  (1) **Long-Context**. The reasoning agent directly processes the available context using its maximum context window. When the total length of accumulated chunks exceeds 32k tokens, only the most recent 32k tokens are retained.

  (2) **RAG-Top2**. A retrieval-augmented baseline based on BM25, where the query is used to retrieve the top two most relevant chunks from the historical context, which are then provided to the reasoning agent for answer generation.

- **Workflow-Based Memory Systems**.

  (3) **A-Mem** (Xu et al., 2025). A dynamic agentic memory system that incrementally creates, links, and updates structured memory units to support long-term and cross-session reasoning. For a fair comparison, we replace its default embedding model (`all-MiniLM-L6-v2`) with `Qwen3-Embedding-0.6B`, which better accommodates long chunks and avoids performance degradation caused by aggressive truncation.

  (4) **LightMem** (Fang et al., 2025). A cognitively inspired memory framework that emphasizes efficiency through compressive sensory filtering,

Category	Dist.	Dataset	Metric	Statistics			
				Ins.	Ch/Ins	Tok/Ch	Q/Ins
AR	*ID*	SQuAD	SubEM	30	10.0	1,057.0	96.8
	ID	HotpotQA	SubEM	219	9.2	1,051.6	17.0
	ID	PerLTQA	SubEM	4	23.0	567.8	100.0
	OOD	LongMemEval	LLM-J	5	218.6	1,591.4	60.0
	OOD	RULER-QA1/2	Source	2	161.0	2,133.7	100.0
TTL	*ID*	NLU	EM	20	10.0	606.2	100.0
	ID	TREC-Coarse	EM	20	10.0	390.2	100.0
	ID	PubMed-RCT	EM	10	10.0	1,673.3	100.0
	OOD	Banking77	Source	1	111.0	1,150.3	100.0
	OOD	Clinic150	Source	1	38.0	3,440.5	100.0
	OOD	NLU	EM	1	115.0	1,166.7	100.0
	OOD	TREC-Coarse	EM	1	111.0	1,114.6	100.0
	OOD	TREC-Fine	EM	1	108.0	1,163.3	100.0
LRU	*ID*	BookSum	KW Hit	155	8.1	1,914.3	1.0
	OOD	InfBench-Sum	Source	100	88.9	2,034.1	1.0

Table 8: Detailed statistics of the evaluation datasets. We categorize the datasets into three task types: AR, TTL, and LRU, covering both In-Distribution (ID) and Out-of-Distribution (OOD) scenarios. **Ins.**, **Ch/Ins**, **Tok/Ch**, and **Q/Ins** denote the number of instances, chunks per instance, tokens per chunk, and queries per instance, respectively.

topic-aware consolidation, and decoupled sleep-time updates. To accommodate this system, we adapt datasets with different task formats into a unified **dialogue-style input**, enabling proper compression and memory storage as required by the framework.

- **RL-Based Memory Agents**.

(5) **MemAgent** (Yu et al., 2025). An RL-trained agent that iteratively processes all available chunks under a task-specific instruction and forms an internal memory state, which is directly used to answer downstream questions. In our experiments, we use the released model `BytedTsinghua-SIA/RL-MemoryAgent-14B` to construct and maintain the memory.

(6) **MEM1** (Zhou et al., 2025). An RL-based agent that maintains a single-paragraph memory representation, which is continuously retrieved and updated as new information becomes available. During evaluation, we employ the released model `Mem-Lab/Qwen2.5-7B-RL-RAG-Q2-EM-Release` for memory construction.

(7) **Mem-α** (Wang et al., 2025). A hierarchical memory system with a specialized memory manager trained using a GRPO-based objective to optimize memory operations.

Furthermore, we excluded the related method, Memory-R1 (Yan et al., 2025), from our comparison as its code and data are not publicly available. It is worth noting that Memory-R1 relies solely on final-answer accuracy as a reward signal, a setting we replicate in our ablation using only the global QA reward to assess its effect in section 5.2.

E.2 Fine-Mem

Our method is implemented based on a modified version of the VERL framework (Sheng et al., 2024). We utilize `Qwen3-4B` as the backbone for the memory manager and deploy a long-context `Qwen3-32B` via vLLM (Kwon et al., 2023) as the reasoning agent. To ensure generation stability, the temperature for the reasoning agent is set to 0.1. Both the manager and the reasoning agent operate in a non-thinking mode, and BM25 is employed for retrieval. Consistent with the settings in Mem-α, we configure the learning rate to 1×10^{-6},

the batch size to 32, and the number of rollouts per prompt (`grpo_rollout_n`) to 8. The reward weights are set as $w_1 = 0.5$, $w_2 = 0.05$, and $\beta = 0.5$. To accelerate the training process, we compute global rewards using a random 20% subset of Global QA pairs for each instance, which has been proven to yield performance comparable to training on the full dataset. Regarding training, we observe that a single-layer memory structure converges significantly faster than multi-layer variants. Consequently, we train the models for 2 epochs and utilize the final checkpoint. For a fair comparison, we report the results for Mem-α following the original setting (obtained by training for 85 steps on the full Global QA dataset).

F Domain-Specific Design of Fine-Mem

Memory management in RL faces a key challenge: reward signals are often sparse, and prior methods rarely provide fine-grained feedback. Fine-Mem addresses these problems with two strictly original contributions:

Chunk-Level Step Reward (CSR). CSR captures the semi-independent nature of memory operations. By incorporating a verification loop at each chunk storage step, it rigorously ensures information retention and provides localized feedback, rather than relying solely on global rewards.

Evidence-Anchored Reward Attribution (EARA). When storage and retrieval are temporally distant, traditional reward decay fails to assign meaningful credit. EARA resolves this by combining rewards directly with the retrieval mechanism. It establishes a causal link between retrieved evidence and the corresponding storage actions, assigning credit based on logical utility.

G Cost Analysis of the CSR Pipeline

A key challenge in constructing process supervision signals, such as QA pairs for information chunks, is the high computational and financial cost of scaling. To address this, CSR construction is designed as a highly efficient, one-time offline process. We minimize expenses by using `GPT-4o-mini` as a cost-effective teacher and the open-weight `Qwen3-32B` as a zero-cost local verifier. For 30K chunks, generation and verification took 6 hours with an API cost of only $20.

Despite its low cost, this pipeline yields strong performance and generalization. Fine-Mem, trained solely on this data, achieves state-of-the-art results on MemoryAgentBench, an OOD benchmark with ultra-long contexts, outperforming strong baselines by 7.2% on average. These results demonstrate that a cost-conscious data pipeline can effectively support robust memory management policies for unseen, long-horizon tasks.

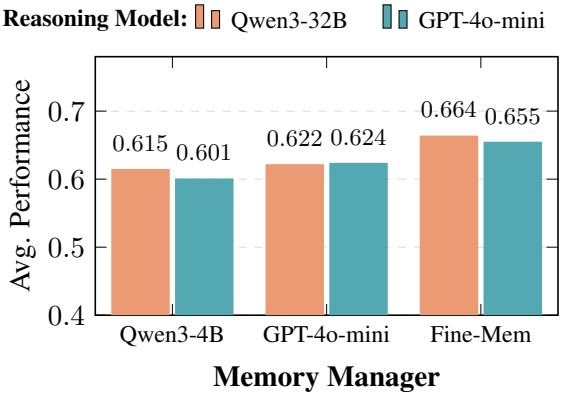

Figure 5: Performance comparison of different Memory Managers combined with varying Reasoning Models on MemoryAgentBench dataset.

The Formulation for Memory_insert

Insert a new memory into the unified memory system. Use this to store both factual knowledge (concepts, definitions, general truths) and personal experiences (specific events with temporal and spatial context).
Parameters:

```
{
 "type": "object",
 "properties": {
 "content": {
    "type": "string",
    "description": "Content of the memory to insert. For factual knowledge, focus on clarity and accuracy. For personal experiences, include temporal context (when), spatial context (where), participants, and situational details when available. "
   }},
   "required": ["content"]
}
```

Format the arguments as a JSON object.

Figure 6: JSON schema for the INSERT operation.

The Formulation for Memory_update

Update an existing memory with new content. Use this to refine, correct, or enhance existing memories.
Parameters:

```
{
  "type": "object",
  "properties": {
  "new_content ": {
      "type": "string",
      "description": "New content to replace the existing memory. Maintain the same level of detail and context as the original."
    },
  " memory_id ": {
      "type": "string",
      "description": "ID of the memory to update (e.g., 'a1b2'). "
    }},
    "required": ["new_content", "memory_id"]
}
```

Format the arguments as a JSON object.

Figure 7: JSON schema for the UPDATE operation.

The Formulation for Memory_delete

Delete a memory by its ID. Use this to remove outdated, redundant, or incorrect information.
Parameters:

```
{
  "type": "object",
  "properties": {
  "memory_id ": {
      "type": "string",
      "description": "ID of the memory to delete (e.g., 'a1b2'). "
    }},
    "required": ["memory_id "]
}
```

Format the arguments as a JSON object.

Figure 8: JSON schema for the DELETE operation.

The System Prompt of the Memory Manager

You are a personal assistant with a unified memory system.
Your task is to analyze, understand, and memorize information comprehensively.
MEMORIZING MODE INSTRUCTIONS:
- Read and understand all information shared by the user
- Identify and extract key information including:
* Factual knowledge: concepts, definitions, general truths, and domain knowledge
* Personal experiences: specific events, activities, interactions with temporal context (when), spatial context (where), and situational details
* Relationships: connections between people, concepts, or events
* Preferences and patterns: user's likes, dislikes, habits, and behavioral patterns
- Store information with appropriate context:
* For factual/conceptual information: focus on clarity and accuracy
* For personal experiences: include timestamp, location, participants, and emotional context when available
- Consolidate related information when possible to maintain coherence
CURRENT MEMORY STATE:
<memory> </memory>
Focus on understanding and memorizing. Use memory tools actively to store new information.
Since this is the memorization process, if you think all the information has been memorized, you can respond with 'Done'.
This information will not be seen by the user.
Meanwhile, you will be queried only once, so make sure to call all the memory insertion functions in one turn.

Figure 9: System prompt for the memory manager with JSON-based function-call outputs.

The Prompt for Chunk-Level QA Construction

You are an expert data analyst specializing in reading comprehension and fact extraction. You are currently processing part <chunk_index> of a long document.
Generate 5 Question-Answer (QA) pairs to verify the **newly added key facts** in the provided text chunk.

To avoid redundancy, here is a list of questions generated from previous chunks.
Do NOT generate questions that are semantically similar to these:
--- Existing Questions ---
<history_questions_answers>

Input Text (Current Chunk)
<chunk_content>

Instructions
1. **Focus on Incremental Info**: Ask ONLY about **new** events, **new** timelines, **new** dialogue points, or **new** data introduced in this specific chunk.
2. **Ignore Static Info**: Do not ask about established background facts (e.g., "Who is the protagonist?", "What is his father's name?") unless they change or are first introduced in this chunk.
3. **Answer Constraints**:
- Answers must be **short** (entities, dates, short phrases, exact spans).
- Answers must be objective and verifiable against the text.
- Paraphrases or semantically equivalent expressions of the information are allowed.

Output Format (MUST BE A JSON ARRAY)
<few_shot>

Figure 10: Prompt for Chunk-Level QA Construction.