MEM-SONAR: SELF-ORGANIZING AND SELF-NAVIGATING MEMORY VIA ALTERNATING REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term memory systems enable large language models (LLMs) to retain and use information beyond their context windows. However, external memory often remains an opaque black box: without prior awareness of how its contents are organized, the LLM must rely on exploratory queries and repeated retrieval attempts. We refer to this problem as *lack of memory priors*. We introduce MEM-SONAR, which models memory organization and retrieval as two trainable capabilities of the same LLM. The Organizer generates Semantic Paths to organize atomic facts into a memory tree, while the Navigator follows these paths to retrieve evidence based on downstream questions. To improve and align memory organization and retrieval, we further propose multi-round alternating reinforcement learning. Retrieval performance provides feedback for improving memory organization, while the resulting structures support subsequent navigation training. Through this bidirectional learning process, the model progressively learns an organization strategy suited to its own retrieval behavior, developing an *implicit memory prior* that guides memory access. Experiments show that MEM-SONAR outperforms the compared baselines across multiple benchmarks and backbones. Further analyses demonstrate sustained improvements in both organization and retrieval, supporting the effectiveness of their bidirectional learning.[1]

1 INTRODUCTION

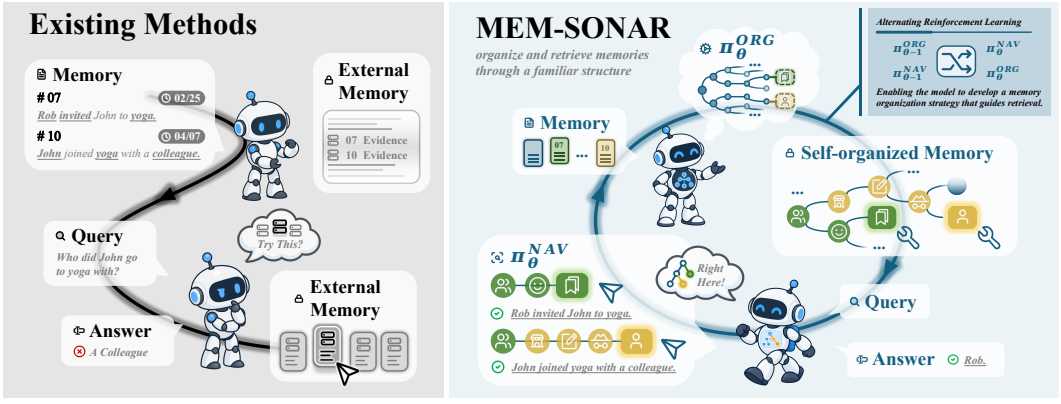

Figure 1: Comparison of existing method and MEM-SONAR.

As large language models (LLMs) evolve from single-turn question-answering systems into agents that continuously perceive, reason, and act, maintaining and utilizing accumulated information becomes increasingly important. This makes long-term memory systems (Hu et al., 2026) an essential component of modern agent architectures. A memory system typically consists of two core components: an LLM and an external memory module. The former performs language understanding

[1] Anonymous code repository: `https://anonymous.4open.science/r/SONAR-DC8E/`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

and reasoning, while the latter is responsible for memory organization and retrieval. Existing studies have advanced both components. MemGPT manages long-term context through hierarchical storage (Packer et al., 2024), Mem0 and A-MEM organize memories through graph-based relations (Chhikara et al., 2025; Xu et al., 2025), and SimpleMem combines semantic compression with query-aware retrieval (Liu et al., 2026). More recently, Memory-R1 and Mem-α introduce reinforcement learning into memory extraction, writing, updating, and selection, enabling the LLM to learn how to use the external memory module from downstream task feedback (Yan et al., 2026b; Wang et al., 2025).

However, existing external memory modules commonly adopt a flat storage paradigm: memories are stored as independent entries, indexed primarily through vector representations, and retrieved largely through vector similarity matching. At inference time, external memory often remains an opaque "black box" to the LLM. Without an overview of how information is distributed across the memory store, the model must rely on exploratory queries through retrieval tools, obtaining only fragments of relevant information. Since the model cannot readily determine what other relevant content remains available, it can become trapped in repeated query reformulation and unguided retrieval attempts. We define this problem of unguided retrieval caused by a lack of prior awareness of memory contents as **lack of memory priors**. An intuitive approach is to let the model autonomously organize memories into a tree, assigning them to hierarchical branches according to its own semantic categorization logic. During retrieval, the same LLM starts at the root and selects branches level by level based on the question and the semantic labels of available branches. Reinforcement learning can then guide the model toward a memory organization strategy suited to its own retrieval behavior. In this way, the model's understanding of its own memory organization strategy constitutes an **implicit memory prior** that guides retrieval.

However, realizing this idea presents two key challenges. First, memory organization involves semantic categorization based on the facts themselves, whereas retrieval requires determining where to search based on a question. The semantic judgments made in these two settings may not align. Consequently, semantically reasonable categorization does not necessarily lead to effective retrieval: **memory organization must accommodate the demands of question-driven retrieval**. Second, the model's retrieval ability constrains the optimization of memory organization. Learning an organization strategy that supports question answering requires using retrieval performance as feedback to identify more effective strategies. However, if the model cannot accurately navigate the current structure based on a question, potentially effective organization strategies may fail to receive positive feedback because of inadequate retrieval ability, leaving the learned organization strategy suboptimal. The key to realizing this idea is therefore to establish a bidirectional learning relationship between memory organization and retrieval. On the one hand, the model learns to retrieve memories more effectively using its own organization strategy; on the other, it uses retrieval performance as feedback to improve that strategy. Through this bidirectional learning process, memory organization and retrieval become progressively better aligned and improve together.

Based on these insights, we propose MEM-SONAR (**S**elf-**O**rganizing and Self-**N**avigating Memory via **A**lternating **R**einforcement Learning), which models memory organization and retrieval as two abilities of a single LLM, realized through the Organizer and Navigator roles, respectively. The two roles share model parameters and jointly maintain and use a memory structure autonomously organized by the model. This structure connects multiple memory trees through a global root node, with internal nodes providing semantic labels at different levels and leaf nodes storing atomic facts. During memory storage, the Organizer extracts atomic facts from the input and generates semantic paths that determine their locations within the memory structure. During retrieval, the Navigator follows the tree structure from the root, selecting branches level by level based on the question and branch labels to locate the required knowledge. This formulation makes memory organization and retrieval trainable decision processes within the same model.

To establish this bidirectional learning relationship, we further propose a multi-round Alternating Reinforcement Learning mechanism. During the Organizer optimization stage, the Organizer generates multiple candidate memory trees, and the Navigator attempts to answer questions using these trees. The resulting question-answering performance serves as the reward signal for optimizing the Organizer, encouraging it to learn organization strategies that better support question answering. During the subsequent Navigator optimization stage, the updated Organizer rebuilds the memory trees, which are then held fixed. The Navigator learns to select paths based on questions within these fixed structures, improving its retrieval ability. The two stages sequentially update the param-

eters of the same underlying LLM. The improved Navigator then participates in the next round of Organizer optimization, providing more effective question-answering feedback for candidate memory trees. Through repeated alternating optimization, better memory organization supports retrieval, while stronger retrieval further improves memory organization. The two abilities thus reinforce each other and improve continuously, more effectively addressing the lack of memory priors. Experiments on long-term memory question answering demonstrate consistent performance gains across different underlying LLMs. Further analyses show that alternating training improves both memory organization and retrieval, supporting the effectiveness of their bidirectional learning.

Our main contributions are as follows:

- We identify **lack of memory priors** as a limitation of existing memory systems that leads to unguided retrieval. To address this limitation, we propose MEM-SONAR, which models memory organization and retrieval as two trainable abilities of a single LLM.
- We introduce multi-round **Alternating Reinforcement Learning**: retrieval feedback improves memory organization, while the resulting memory trees support retrieval training. This bidirectional learning helps the model develop an **implicit memory prior** for question-driven retrieval.
- Experiments on three benchmarks demonstrate that MEM-SONAR achieves state-of-the-art question-answering performance, with consistent gains across different underlying LLMs. Further analyses support the effectiveness of bidirectional learning.

2 RELATED WORK

Long-Term Memory Systems for LLM Agents. Long-term memory systems extend LLM agents beyond finite context windows by storing information for reuse across interactions. MemGPT manages hierarchical memory tiers (Packer et al., 2024), while Mem0 and MemoryOS support memory updating and retrieval (Chhikara et al., 2025; Kang et al., 2025). RAG and SimpleMem provide retrieval-based access to external knowledge and compact memory representations, respectively (Lewis et al., 2020; Liu et al., 2026). LightMem reduces memory-management overhead through staged processing and consolidation (Fang et al., 2026), while GAM combines memory construction with iterative search (Yan et al., 2025). However, efficient storage and retrieval alone do not guarantee the LLM an implicit memory prior.

Hierarchical and Structured Memory. Structured memory provides relations and abstraction levels beyond independent entries. Zep, A-Mem, and AdaMem organize information through temporal knowledge graphs, linked notes, or heterogeneous memory components (Rasmussen et al., 2025; Xu et al., 2025; Yan et al., 2026a). Tree-based approaches further support access at multiple levels of granularity. RAPTOR recursively clusters and summarizes text to construct a hierarchy for retrieval at multiple abstraction levels, while MemTree dynamically organizes incoming information into a memory tree using semantic similarity (Sarthi et al., 2024; Rezazadeh et al., 2025). Structured storage, however, does not by itself align organization with question-driven navigation. Our Organizer generates Semantic Paths that assign atomic facts to a persistent memory tree, and the Navigator follows its branch labels to locate evidence. Both roles share an underlying LLM, allowing task-driven training to align these two capabilities.

Reinforcement Learning for Memory Agents. Recent studies have explored reinforcement learning for memory management and access. Memory-R1 learns policies for memory management and utilization, while Mem-α optimizes memory construction using downstream QA rewards (Yan et al., 2026b; Wang et al., 2025). RMM adapts memory retrieval through online reinforcement learning guided by cited evidence (Tan et al., 2025). Other approaches learn compact memory updates and reasoning over compressed contexts for long-horizon interactions (Yu et al., 2026; Zhou et al., 2026; Wu et al., 2026), while Fine-Mem introduces fine-grained feedback for memory operations (Ma et al., 2026). These studies motivate learning memory formation and access from task feedback. Our work couples these capabilities through a persistent semantic memory tree and multi-round alternating reinforcement learning: navigation outcomes guide fact organization, and updated trees support subsequent navigation training. This bidirectional process aligns organization with evidence seeking, helping the shared model develop an implicit memory prior for question-driven retrieval.

3 METHOD

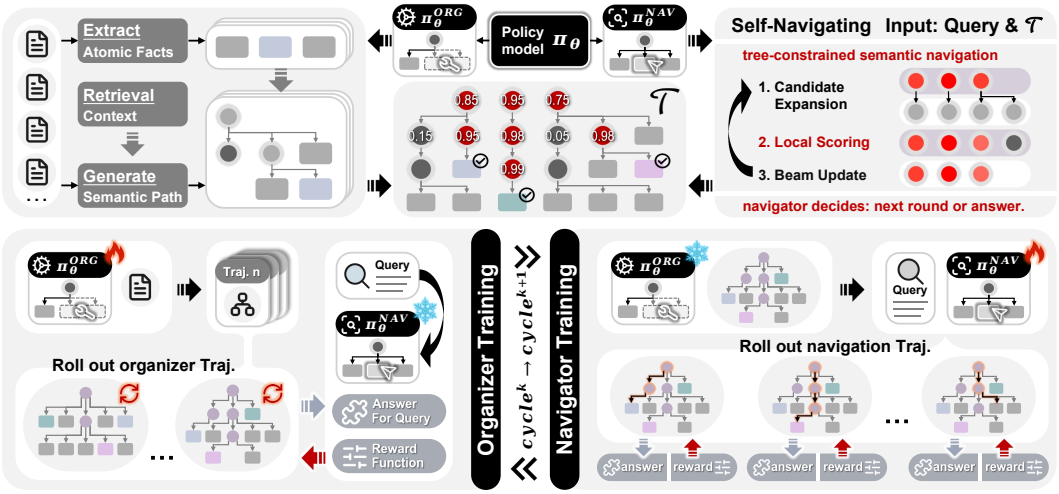

Figure 2: Overview of MEM-SONAR, our self-organizing and self-navigating long-term memory framework with multi-round alternating reinforcement learning.

We propose MEM-SONAR, which integrates memory formation and memory access into the LLM as two executable and learnable capabilities. Specifically, MEM-SONAR instantiates an Organizer and a Navigator from the same underlying language model. The Organizer performs memory formation by extracting incoming facts and generating Semantic Paths that determine their positions in the memory tree, while the Navigator performs memory access by directly navigating the same tree to collect relevant evidence for downstream tasks. The memory tree serves as the persistent storage substrate, whereas its organization and access are controlled by the LLM through these two roles. We further introduce multi-round alternating reinforcement learning to improve organization through retrieval feedback and train retrieval on the updated structures, establishing a bidirectional learning relationship between the two capabilities.

3.1 SELF-ORGANIZING MEMORY VIA SEMANTIC PATHS

We instantiate memory formation as the Organizer's self-organizing capability. The memory structure consists of a global root connecting Episode-level trees, with semantic labels on internal nodes and atomic facts at the leaves. Within each Episode, the Organizer directly determines fact placement by generating natural-language Semantic Paths. Each path specifies a hierarchical location in the local tree, allowing the model to organize incoming facts according to its own semantic categorization.

We partition the incoming data stream $\mathcal{D} = (E^{(1)}, \ldots, E^{(N)})$ into Episodes at natural boundaries, such as topic shifts, temporal gaps, or task completion. Within each Episode $E^{(i)}$, the Organizer processes incoming data items sequentially and extracts atomic facts $f_1^{(i)}, f_2^{(i)}, \ldots$, indexed in processing order. For each $f_j^{(i)}$, the Organizer retrieves context from the current partial Episode tree:

$$c_j^{(i)} \sim \pi_\theta^{\mathrm{NAV}} \left(\cdot \mid f_j^{(i)}, \mathcal{T}^{(i)} \right), \tag{1}$$

Here, $\mathcal{T}^{(i)}$ denotes the current partial tree of Episode $E^{(i)}$, updated after each fact insertion. The Navigator policy $\pi_\theta^{\mathrm{NAV}}$ uses $f_j^{(i)}$ as the query to retrieve context $c_j^{(i)}$ containing relevant semantic nodes and stored facts (Section 3.2).

Rather than decomposing memory formation into a sequence of low-level tree operations, we formulate it as the direct prediction of a complete Semantic Path. For each fact, the Organizer predicts a path from coarse semantic concepts to its target position in the current Episode tree, thereby making a single explicit formation decision. This formulation focuses learning on where a memory should

be organized and shortens the decision chain between model output and the resulting structure, allowing downstream feedback to more directly optimize memory formation. Based on the fact and the retrieved context, the Organizer generates the Semantic Path as its formation action:

$$p_j^{(i)} \sim \pi_\theta^{\mathrm{ORG}}\left(\cdot \mid f_j^{(i)}, c_j^{(i)}\right), \qquad p_j^{(i)} = (z_1, \ldots, z_L). \tag{2}$$

Each z_l is a natural-language semantic label, and $p_j^{(i)}$ specifies the hierarchical position of $f_j^{(i)}$ in the Episode tree. The environment deterministically extends the tree along $p_j^{(i)}$ and inserts $f_j^{(i)}$ at the corresponding leaf, updating $\mathcal{T}^{(i)}$.

Formally, we represent the memory structure as:

$$\mathcal{T} = \left(r, \{\mathcal{T}^{(i)}\}_{i=1}^{N}\right), \qquad \mathcal{T}^{(i)} = \left(V^{(i)}, A^{(i)}, \mathcal{M}^{(i)}\right). \tag{3}$$

Here, r is the global root used for memory navigation. Each Episode tree contains semantic nodes $V^{(i)}$, parent–child relations $A^{(i)}$, and stored atomic facts $\mathcal{M}^{(i)}$. Episode nodes are associated with coarse summaries to support cross-Episode routing, while internal nodes provide semantic routing labels and leaf nodes store factual memories.

The Organizer repeatedly reads the current structure, generates Semantic Paths, and organizes newly extracted facts. This incremental process enables memory self-organization by placing fact organization within the model's learnable decision space.

3.2 Self-Navigating Memory via Tree-Constrained Semantic Navigation

Given a downstream question q, each round begins with the Navigator generating a reformulated query $\tilde{q}$ based on q and any previously collected evidence. The Navigator then routes from the global root to candidate Episodes and follows Semantic Paths within their trees. Each navigation round uses tree-constrained beam search, which iteratively performs *Candidate Expansion*, *Local Scoring*, and *Beam Update* to locate relevant memories.

Search starts at the global root. At step h, *Candidate Expansion* identifies the available child nodes at the endpoint of each path in $\mathcal{B}_{h-1}$. In *Local Scoring*, the Navigator uses the query, current path prefix, and available child labels to propose up to k child-node extensions for each path and generate a relevance score for each:

$$s_h(z_h \mid p_{<h}) = g_\theta^{\mathrm{NAV}}\left(z_h;\, \tilde{q},\, p_{<h},\, \mathcal{O}_{\mathcal{T}}(p_{<h})\right), \qquad p_{<h} = (z_1, \ldots, z_{h-1}). \tag{4}$$

Here, g_θ^{NAV} denotes the LLM's relevance scoring function with outputs in $[0, 1]$, and k denotes Beam Top-k, the per-node expansion limit. The prefix $p_{<h}$ starts at the global root, and $\mathcal{O}_{\mathcal{T}}(p_{<h})$ denotes the local observation at its endpoint. Organization paths $p_j^{(i)}$ start at their Episode roots. Finally, *Beam Update* pools the proposed extensions across all paths and ranks them by the product of their local relevance scores. TopK retains up to K highest-scoring paths as the next beam, where K denotes Beam Width:

$$S(p_{\le h}) = \prod_{t=1}^{h} s_t(z_t \mid p_{<t}), \qquad \mathcal{B}_h = \mathrm{TopK}_{p_{\le h}} S(p_{\le h}). \tag{5}$$

These steps repeat down the tree to complete the current search and collect relevant evidence. At the end of each navigation round, $\pi_\theta^{\mathrm{NAV}}$ either generates the final answer or starts another round based on q and the collected evidence.

3.3 Multi-Round Alternating Reinforcement Learning for Coupled Memory Organization and Navigation

Effective memory organization depends on feedback from question-driven retrieval, whose quality in turn depends on the Navigator's ability to use the current tree. We therefore adopt multi-round alternating reinforcement learning to improve both capabilities. Each cycle first optimizes the Organizer using question-answering rewards on candidate trees. The updated Organizer then reconstructs

the memory trees, which remain fixed during Navigator training. Both stages sequentially update the same underlying LLM, and the resulting Navigator checkpoint initializes the next Organizer stage. Improved navigation then provides more informative feedback on which organization decisions help the model locate relevant evidence. This cycle connects the organization of stored facts with the navigation behavior used to access them, helping the model develop an implicit memory prior for question-driven retrieval.

Organizer optimization. At round k, the Organizer is initialized from the previous Navigator checkpoint $\theta_{k-1}^{\mathrm{NAV}}$. During this stage, both roles share the current parameters θ, updated through Organizer training. For each Episode $E^{(i)}$, we sample G_{ORG} organization trajectories from the same empty tree. Each trajectory $\tau_{\mathrm{ORG},g}^{(k)}$ contains fact extraction and Semantic Path generation decisions and produces a candidate Episode tree, where g indexes sampled trajectories.

Since navigation decisions rely on local tree observations, we apply the same navigation mechanism within each candidate Episode subtree to obtain local feedback for memory organization. The Navigator answers reward questions $\mathcal{Q}^{(i)}$ whose supporting evidence is contained in the rollout tree, yielding the organization reward:

$$r_{\mathrm{ORG}}\left(\tau_{\mathrm{ORG},g}^{(k)};\theta\right)=\mathrm{F1}\left(\mathcal{Q}^{(i)},\theta,\tau_{\mathrm{ORG},g}^{(k)}\right)+\lambda_{\mathrm{fmt}}R_{\mathrm{fmt}}\left(\tau_{\mathrm{ORG},g}^{(k)}\right). \tag{6}$$

Here, θ denotes the shared parameters at evaluation time. F1 is averaged over $\mathcal{Q}^{(i)}$, and R_{fmt} rewards valid extraction and path formats with weight λ_{fmt}. Retrieval feedback thus guides the Organizer toward organization strategies suited to question-driven retrieval.

Navigator optimization. The updated Organizer with parameters θ_k^{ORG} reconstructs the memory tree $\mathcal{T}_k$ from $\mathcal{D}$. We initialize the Navigator from θ_k^{ORG} and optimize it on $\mathcal{T}_k$, which remains fixed throughout this stage. For each question q, we sample G_{NAV} trajectories $\tau_{\mathrm{NAV},g}^{(k)}$, each containing tree traversal, multi-round navigation decisions, and final answer generation. The reward is:

$$r_{\mathrm{NAV}}\left(\tau_{\mathrm{NAV},g}^{(k)}\right)=\mathrm{F1}\left(q,\tau_{\mathrm{NAV},g}^{(k)}\right). \tag{7}$$

F1 measures agreement between the final answer and the reference for q, guiding the Navigator to use the existing organization to locate the correct evidence based on the question.

Both stages use Group Relative Policy Optimization (GRPO) (Shao et al., 2024). Let θ_0 be the initial parameters, and θ_k^{ORG} and θ_k^{NAV} the checkpoints after the two stages of round k. With $\theta_0^{\mathrm{NAV}}=\theta_0$, the updates are:

$$\theta_k^{\mathrm{ORG}}=\mathrm{GRPO}_{\mathrm{ORG}}\left(\theta_{k-1}^{\mathrm{NAV}};r_{\mathrm{ORG}}\right),\qquad \theta_k^{\mathrm{NAV}}=\mathrm{GRPO}_{\mathrm{NAV}}\left(\theta_k^{\mathrm{ORG}};\mathcal{T}_k,r_{\mathrm{NAV}}\right). \tag{8}$$

The resulting θ_k^{NAV} initializes the next Organizer stage, carrying retrieval experience into subsequent construction and progressively developing an implicit memory prior that guides retrieval through bidirectional learning.

4 Experiments

4.1 Experimental Settings

We evaluate Mem-Sonar on LoCoMo (Maharana et al., 2024), comparing it with RAG (Lewis et al., 2020), MemGPT (Packer et al., 2024), Mem0 (Chhikara et al., 2025), SimpleMem (Liu et al., 2026), LightMem (Fang et al., 2026), GAM (Yan et al., 2025), and Mem-α (Wang et al., 2025). All baselines and Mem-Sonar use the same test conversations, backbones, and evaluation protocol. We additionally report the published results of Memory-R1-GRPO (Yan et al., 2026b) with LLaMA3.1-8B (Grattafiori et al., 2024) and Qwen2.5-7B (Qwen Team, 2025), marked with † to distinguish them from locally reproduced results. We further evaluate generalization on LongMemEval (Wu et al., 2025) and NarrativeQA (Kočiský et al., 2018). Our main backbones are Qwen3-4B-Instruct-2507 (Yang et al., 2025) and Qwen2.5-7B-Instruct (Qwen Team, 2025). We report token-level F1 and BLEU-1 (B1) (Papineni et al., 2002) on a 0–100 scale. Detailed experimental settings are provided in Appendix A.

Table 1: Performance comparison on the LoCoMo benchmark using F1 and BLEU-1. Bold and underlined values denote the best and second-best results within each locally evaluated backbone setting, respectively. †Results are reproduced from the original paper for reference.

Method	Base LLM	Single-Hop		Multi-Hop		Temporal		Open-Domain		Overall	
		F1↑	B1↑	F1↑	B1↑	F1↑	B1↑	F1↑	B1↑	F1↑	B1↑
RAG (Lewis et al., 2020)	Qwen3-4B	44.94	40.20	24.59	18.63	29.84	25.75	13.17	9.00	36.24	31.40
MemGPT (Packer et al., 2024)	Qwen3-4B	15.13	12.98	15.49	12.26	9.58	8.12	16.37	14.41	14.05	11.88
Mem0 (Chhikara et al., 2025)	Qwen3-4B	34.64	31.61	21.07	15.19	26.15	22.52	16.63	12.74	29.47	25.65
SimpleMem (Liu et al., 2026)	Qwen3-4B	39.78	35.40	28.75	21.27	42.84	35.40	22.04	17.95	37.77	32.12
Mem-α (Wang et al., 2025)	Qwen3-4B	39.02	34.55	28.52	20.96	25.58	21.65	12.79	9.23	32.97	28.07
LightMem (Fang et al., 2026)	Qwen3-4B	41.75	36.56	32.83	25.55	36.97	30.35	11.71	8.2	37.89	32.08
GAM (Yan et al., 2025)	Qwen3-4B	<u>54.44</u>	<u>49.54</u>	<u>41.30</u>	<u>33.60</u>	<u>49.23</u>	<u>43.06</u>	<u>23.56</u>	<u>20.10</u>	<u>49.67</u>	<u>44.03</u>
MEM-SONAR	Qwen3-4B	**63.32**	**57.55**	**43.07**	**35.05**	**72.63**	**68.14**	**27.30**	**23.76**	**58.86**	**53.53**
Memory-R1† (Yan et al., 2026b)	LLaMA3.1-8B	35.73	27.70	35.65	30.77	49.86	38.27	47.42	41.24	45.02	37.51
Memory-R1† (Yan et al., 2026b)	Qwen2.5-7B	33.64	26.06	23.55	20.71	47.75	38.49	46.86	40.92	43.14	36.44
RAG (Lewis et al., 2020)	Qwen2.5-7B	37.43	33.47	21.91	16.10	27.68	23.51	14.79	11.94	31.63	27.23
MemGPT (Packer et al., 2024)	Qwen2.5-7B	17.78	13.61	10.37	8.17	9.74	7.51	9.35	6.76	14.43	11.13
Mem0 (Chhikara et al., 2025)	Qwen2.5-7B	36.15	32.11	20.48	14.17	25.50	21.44	11.46	8.83	29.77	25.25
SimpleMem (Liu et al., 2026)	Qwen2.5-7B	39.39	35.40	28.41	20.95	38.76	32.36	22.69	18.76	36.64	31.47
Mem-α (Wang et al., 2025)	Qwen2.5-7B	38.88	34.83	30.60	21.68	36.84	32.15	17.02	14.66	36.19	31.08
LightMem (Fang et al., 2026)	Qwen2.5-7B	46.23	40.37	32.63	23.43	<u>43.32</u>	<u>38.06</u>	22.78	18.93	42.35	36.02
GAM (Yan et al., 2025)	Qwen2.5-7B	<u>53.85</u>	<u>48.86</u>	<u>34.86</u>	<u>27.92</u>	38.87	32.93	<u>25.34</u>	<u>21.75</u>	<u>45.58</u>	<u>39.95</u>
MEM-SONAR	Qwen2.5-7B	**56.21**	**51.22**	**41.07**	**31.89**	**59.62**	**54.00**	**31.02**	**27.22**	**52.75**	**46.86**

4.2 OVERALL PERFORMANCE

Table 1 shows that MEM-SONAR achieves the highest F1 and BLEU-1 among locally evaluated methods across both backbones and all four question types. Compared with the strongest baseline, GAM, it improves overall F1 by 9.19 points with Qwen3-4B and 7.17 points with Qwen2.5-7B. Moreover, MEM-SONAR with Qwen3-4B surpasses the reported overall F1 and BLEU-1 of Memory-R1 with the larger LLaMA3.1-8B and Qwen2.5-7B models, demonstrating its competitiveness with a smaller backbone. Beyond LoCoMo, MEM-SONAR with Qwen3-4B also ranks first on both LongMemEval and NarrativeQA (Table 2). These results demonstrate the performance advantages of MEM-SONAR across different backbones and question-answering tasks.

Table 2: Generalization results on additional benchmarks.

Method	LongMemEval	NarrativeQA	
	Acc	F1	BLEU-1
GAM	<u>54.20</u>	<u>32.09</u>	<u>27.68</u>
LightMem	47.80	19.99	17.33
MemGPT	38.20	20.86	15.46
Mem-α	27.20	24.51	20.75
SimpleMem	28.80	26.66	23.21
RAG (BM25)	54.20	12.17	9.09
MEM-SONAR (Ours)	**69.40**	**35.26**	**31.26**

4.3 BIDIRECTIONAL LEARNING OF MEMORY ORGANIZATION AND RETRIEVAL

We examine whether bidirectional learning improves memory organization and retrieval over successive training cycles. Figure 3(a) tracks overall F1 across Cycles 1–9. The dashed curve uses each cycle's final model, after Navigator training, for both memory organization and retrieval. Performance improves in every cycle and continued progress in later cycles. These sustained gains support the benefit of repeatedly using retrieval feedback to refine organization and training retrieval on the resulting structures. To examine how each ability contributes, we pair Organizer and Navigator checkpoints from different cycles, shown by the solid curves in Figure 3(a). With the Navigator fixed, later Organizer checkpoints consistently yield higher F1, indicating that the learned organization better supports retrieval. With the Organizer fixed, later Navigator checkpoints generally improve F1 on the same trees, showing that the model also learns to use an existing organization more effectively. Together, these trends support the bidirectional learning proposed in our framework: memory organization becomes more suitable for retrieval, while retrieval becomes more effective at using the learned organization.

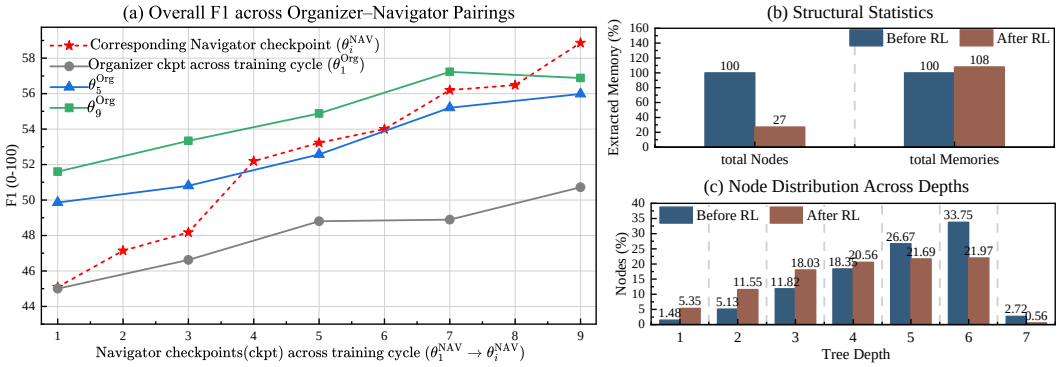

Figure 3: Optimization and memory structure. (a) Solid curves fix the Organizer checkpoint and vary the Navigator cycle. The dashed Corresponding Navigator checkpoint curve uses each cycle's final model for both tree construction and retrieval. (b) Node and memory counts normalized to their respective before-training values (100%). (c) Depth distributions use total nodes, including the unplotted root, as the denominator. Both structural panels share the same color key.

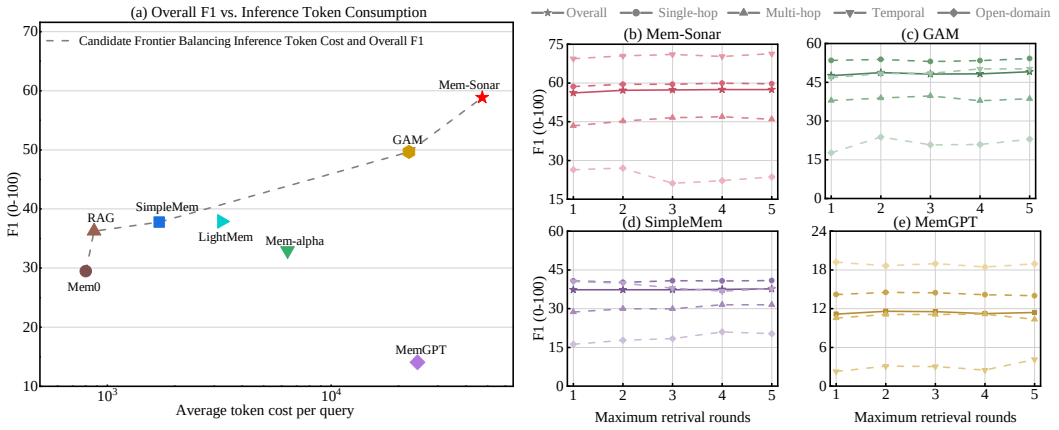

Figure 4: Inference cost and retrieval-round sensitivity on LoCoMo with Qwen3-4B. (a) Overall F1 versus average inference tokens per question, with the candidate frontier. (b–e) Overall and category-level F1 under maximum retrieval-round limits of 1–5 for MEM-SONAR and 3 baseline.

Training also reshapes the memory organization itself. Figure 3(b–c) shows relative changes in node and memory counts, alongside node depth distributions before and after training. The number of structural nodes decreases substantially, while the number of stored facts increases modestly. The depth distribution becomes less concentrated at deeper levels and more balanced across intermediate depths. These changes suggest that reinforcement learning, guided by downstream performance, enables the model to discover a more compact memory organization with a more balanced depth distribution. Together with the checkpoint pairing results, these findings support our insight that task feedback can shape memory organization while the model learns to use it for question-driven retrieval. Additional cross-model evaluations without further training (Appendix A.6) show that using the same backbone for organization and retrieval does not consistently yield the best performance. This is consistent with our analysis, supporting our motivation to develop useful implicit memory priors through bidirectional learning between organization and retrieval.

4.4 INFERENCE COST AND RETRIEVAL-ROUND SENSITIVITY

We examine the performance–cost trade-off on LoCoMo with Qwen3-4B, comparing overall F1 against average inference tokens per question. Measurement details are provided in Appendix A.3. Figure 4(a) shows that MEM-SONAR achieves the highest overall F1 among the compared methods, occupying the high-performance end of the displayed candidate frontier at a higher inference cost.

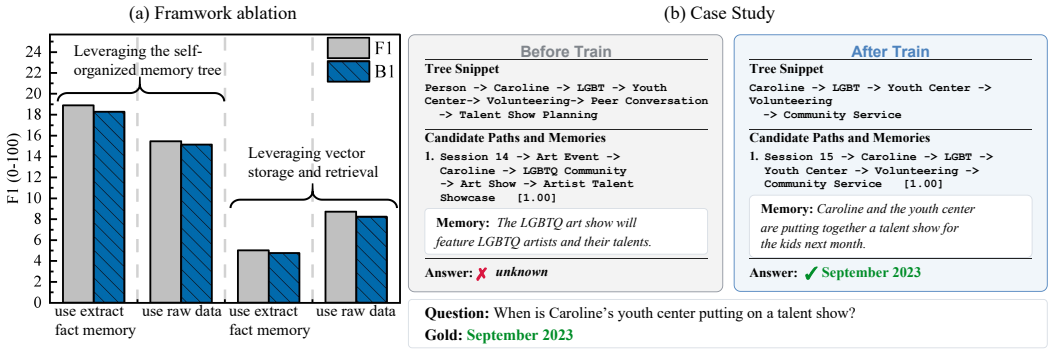

Figure 5: Framework ablation and case study. (a) F1 and BLEU-1 training gains with self-organized memory trees or vector storage and retrieval, each using extracted facts or raw content. (b) Memory organization and retrieved evidence before and after training for the same question.

To assess whether a larger retrieval budget improves performance, we increase the maximum retrieval rounds from one to five in Figure 4(b–e). Overall F1 largely plateaus within the first few rounds, and MEM-SONAR maintains its performance advantage throughout. These results indicate that allowing more retrieval rounds does not consistently yield better answers. This observation is consistent with our motivation that a lack of memory priors can limit the benefits of a larger retrieval budget. Viewed alongside Figure 4(a), the advantage of MEM-SONAR supports the value of bidirectional learning between memory organization and retrieval: the learned organization can serve as an implicit memory prior that guides more effective retrieval, despite the additional cost of LLM-generated navigation decisions.

4.5 FRAMEWORK ABLATION AND CASE STUDY

We ablate memory content and retrieval structure to examine their contributions to the gains from training. Figure 5(a) reports F1 and BLEU-1 gains after training for self-organized memory trees and vector storage and retrieval, each using extracted atomic facts or raw content. Memory-tree variants achieve larger gains with both types of content. Using atomic facts with the memory tree yields the largest gains in both metrics, including an 18.88-point F1 increase compared with 5.02 points under vector retrieval. Within the tree setting, atomic facts also outperform raw content in training gains. These results support making memory organization and retrieval jointly learnable, with atomic facts providing effective memory units for this design.

Figure 5(b) illustrates how the learned organization and retrieval work together on a concrete question. Before training, the model follows an Art Show branch in Session 14 when asked about Caroline's youth-center talent show, retrieves irrelevant evidence, and answers *unknown*. After training, it follows the youth-center branch in Session 15 and retrieves the memory that the show will take place *next month*. Combining this evidence with the session date yields the correct answer, *September 2023*. This example illustrates how the model can use its learned organization to connect a question with the relevant memory, rather than being misled by a superficially similar event.

5 CONCLUSION

We presented MEM-SONAR, which addresses lack of memory priors by learning memory organization and retrieval within a single LLM. Through multi-round alternating reinforcement learning, the model learns to organize memories for retrieval and to retrieve evidence using its evolving organization strategy. Experiments demonstrate consistent question-answering improvements across benchmarks and backbones, while further analyses show gains in both abilities and changes in the learned memory structures. These findings support bidirectional learning as a way to develop implicit memory priors that guide question-driven retrieval.

AI USE STATEMENT

We used generative AI tools to assist with drafting and translating manuscript sections, polishing language, and checking grammar and formatting. The authors reviewed and revised the AI-assisted text and take responsibility for the final content.

ETHICS STATEMENT

Our experiments use only publicly available research benchmarks and do not involve the collection of new data from human participants. The datasets used in this work do not contain private or sensitive personal information requiring special handling. Our study is conducted solely in a research setting and does not involve deployment to end users. Therefore, we do not identify additional ethical concerns specific to the experiments conducted in this work.

REPRODUCIBILITY STATEMENT

Section 3 specifies the memory representation, Organizer and Navigator procedures, reward functions, and alternating optimization objective. Section 4 describes the datasets, baselines, evaluation metrics, and reported analyses. Appendix A provides the dataset construction details, training procedure, hyperparameters, inference and analysis configuration, and computing infrastructure needed to reproduce the experiments.

REFERENCES

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=dyJ0GWpjJB.

Aaron Grattafiori et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the age of ai agents, 2026. URL https://arxiv.org/abs/2512.13564.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent, 2025. URL https://arxiv.org/abs/2506.06326.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023/.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. Association for Computing Machinery, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. SimpleMem: Efficient lifelong memory for LLM agents. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=oBgLvd5YC6.

Weitao Ma, Xiaocheng Feng, Lei Huang, Xiachong Feng, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Bing Qin. Fine-Mem: Fine-grained feedback alignment for long-horizon memory management. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19666–19684. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.900. URL https://aclanthology.org/2026.acl-long.900/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318. Association for Computational Linguistics, 2002. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

Qwen Team. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. URL https://arxiv.org/abs/2501.13956.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for LLMs. In *International Conference on Learning Representations*, volume 2025, pp. 990–1023, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/0382cb76309820f71c6eacd47b36ce71-Abstract-Conference.html.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, volume 2024, pp. 32628–32649, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8a2acd174940dbca361a6398a4f9df91-Abstract-Conference.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.413. URL https://aclanthology.org/2025.acl-long.413/.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning, 2025. URL https://arxiv.org/abs/2509.25911.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *Proceedings of the International Conference on Learning Representations (ICLR)*, volume 2025, pp. 86809–86836, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d813d324dbf0598bbdc9c8e79740ed01-Paper-Conference.pdf.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. ReSum: Unlocking long-horizon search intelligence via context summarization, 2026. URL https://arxiv.org/abs/2509.13313.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 17577–17604. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/19909c36f51abc4856b4560aff3d36d6-Paper-Conference.pdf.

B. Y. Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General agentic memory via deep research, 2025. URL https://arxiv.org/abs/2511.18423.

Shannan Yan, Jingchen Ni, Leqi Zheng, Jiajun Zhang, Peixi Wu, Dacheng Yin, Jing Lyu, Chun Yuan, and Fengyun Rao. AdaMem: Adaptive user-centric memory for long-horizon dialogue agents, 2026a. URL https://arxiv.org/abs/2603.16496.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825. Association for Computational Linguistics, 2026b. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

An Yang et al. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=k5nIOvYGCL.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583. Curran Associates, Inc., 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/724be4472168f31ba1c9ac630f15dec8-Paper-Conference.pdf.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=XY8AaxDSLb.

A Additional Experimental Settings

A.1 Dataset Details

LoCoMo. LoCoMo (Maharana et al., 2024) is a benchmark for evaluating long-term conversational memory and reasoning over interactions distributed across multiple sessions. The evaluation questions cover Single-Hop, Multi-Hop, Temporal, and Open-Domain categories, requiring models to access and integrate information distributed throughout long interaction histories. The LoCoMo data used in our experiments contain 10 conversations, comprising 272 dialogue sessions, 5,882 dialogue turns, and 1,986 question–answer pairs. Of the 1,986 question–answer pairs, 1,540 are non-adversarial questions in Categories 1–4, including 841 Single-Hop, 282 Multi-Hop, 321 Temporal, and 96 Open-Domain questions. The remaining 446 questions belong to the adversarial Category 5 and are excluded from the final evaluation.

All locally evaluated baselines and Mem-Sonar use the same evaluation protocol. Conversations 26, 41, 47, and 48 are used for training, while conversations 30, 42, 43, 44, 49, and 50 are used for evaluation. The six evaluation conversations contain 895 non-adversarial questions in Categories 1–4, including 484 Single-Hop, 178 Multi-Hop, 181 Temporal, and 52 Open-Domain questions. The main results are aggregated over conversations 30, 42, 43, 44, 49, and 50 using conversation-level macro averaging. Since conversation 30 contains no Open-Domain questions, it is excluded only when computing the Open-Domain metrics. Overall scores are obtained by directly averaging the per-conversation Overall scores rather than recomputing them from the question categories.

LongMemEval. LongMemEval (Wu et al., 2025) evaluates memory over long-term user–assistant interactions, requiring models to recover and integrate information distributed across multiple sessions. We use it to examine whether the memory organization and access capabilities learned by Mem-Sonar transfer beyond the LoCoMo conversational setting.

NarrativeQA. NarrativeQA (Kočiský et al., 2018) evaluates reading comprehension over long narrative documents, where answering questions often requires integrating information distributed across distant parts of the text. We use it to examine long-range memory organization and evidence access in long-document settings.

For LongMemEval and NarrativeQA, we evaluate baselines using their official implementations and recommended settings. We use officially released models when available and otherwise use the local open-source backbone, with identical hardware and scoring pipelines. Methods without a released implementation or checkpoint are excluded.

A.2 Alternating Training Details

Overall training procedure. Mem-Sonar is trained using multi-round Organizer–Navigator alternating optimization. Each training cycle contains an Organizer optimization stage followed by a Navigator optimization stage. After Organizer optimization, the updated Organizer reconstructs the memory trees from the original interaction histories, and the resulting memory environment is kept fixed during the corresponding Navigator stage. After Navigator optimization, the updated model parameters are used to initialize the Organizer stage of the next cycle. We perform nine alternating cycles in total. Both the Organizer and Navigator are trained for one epoch in each cycle, and the checkpoints from all cycles are retained for subsequent analysis.

Both stages are optimized with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). Model-generated fact extraction, Semantic Path generation, navigation decisions, re-retrieval control, and answer generation constitute trainable model outputs, whereas tree insertion, Beam-state maintenance, memory collection, and deduplication are executed deterministically by the environment.

Organizer training. The Organizer uses one dialogue session as its basic rollout unit. Each optimization step samples two sessions and generates eight independent memory-construction trajectories for each session, resulting in 16 trajectories per rollout step.

Table 3: Main training configurations for the Organizer and Navigator.

Configuration	Organizer	Navigator
Epochs per cycle	1	1
GRPO group size	8 / session	8 / question
Instances per rollout step	2 sessions	4 questions
Trajectories per rollout step	16	32
Learning rate	1×10^{-6}	1×10^{-6}
Weight decay	0.1	0.1
GRPO clipping range	[0.20, 0.28]	[0.20, 0.28]
KL coefficient	0.02	0
Temperature	0.8	0.8
Top-p	0.95	0.95

Within each dialogue session, interactions are processed chronologically, with every four consecutive dialogue turns grouped into one memory-construction segment. The preceding segment is provided as local context to facilitate cross-turn reference and temporal understanding, while information already contained in the preceding segment is not treated as a new extraction target. At most 10 factual memories are retained from each segment.

Extracted facts are organized sequentially. Before generating the Semantic Path for the current fact, the Organizer reads the partial memory tree constructed from previously processed facts. Once organized, the current fact is immediately inserted into the tree before the next fact is processed.

Fact extraction and Semantic Path generation use a sampling temperature of 0.8 and top-$p = 0.95$. The maximum generation length for both operations is 1,024 tokens. Organizer-side partial-tree navigation uses Beam Width $K = 3$ and Beam Top-k $k = 3$.

Navigator training. The Navigator is trained on the memory trees reconstructed by the current Organizer and kept fixed throughout the corresponding Navigator stage. Each question forms an independent GRPO group. A rollout step contains four questions, with eight navigation trajectories sampled for each question, resulting in 32 trajectories per rollout step.

Navigator rollouts use a sampling temperature of 0.8 and top-$p = 0.95$. The maximum output lengths for Beam node selection, re-retrieval control, and final answer generation are 400, 300, and 200 tokens, respectively. Training-time navigation uses Beam Width $K = 5$ and Beam Top-k $k = 3$.

Optimization hyperparameters. Both stages use the Adam optimizer (Kingma & Ba, 2015) with a constant learning rate of 1×10^{-6}, weight decay of 0.1, and $(\beta_1, \beta_2) = (0.9, 0.98)$. The lower and upper GRPO clipping thresholds are set to 0.20 and 0.28, respectively, and the entropy coefficient is set to 0. The KL coefficients are 0.02 for the Organizer and 0 for the Navigator.

Table 3 summarizes the main training hyperparameters.

A.3 Inference and Analysis Configuration

Inference configuration. At evaluation time, a separate memory tree is constructed for each conversation. Different sessions may be processed in parallel, while fact extraction and incremental organization within the same session follow the original chronological order. Consistent with training, every four consecutive dialogue turns form one interaction segment.

Question-time navigation uses Beam Width $K = 9$ and Beam Top-k $k = 9$. Local relevance scores are floored at 10^{-3} before computing their path product. The Navigator performs at most three retrieval or re-retrieval rounds. Organizer-side partial-tree navigation and question-time navigation follow the same tree-constrained navigation mechanism described in the main paper, while using their respective search budgets. All evaluation-time model calls use greedy decoding.

All locally evaluated methods use the same answer normalization and metric implementation. Before computing F1 and BLEU-1 (Papineni et al., 2002), predictions and reference answers are low-

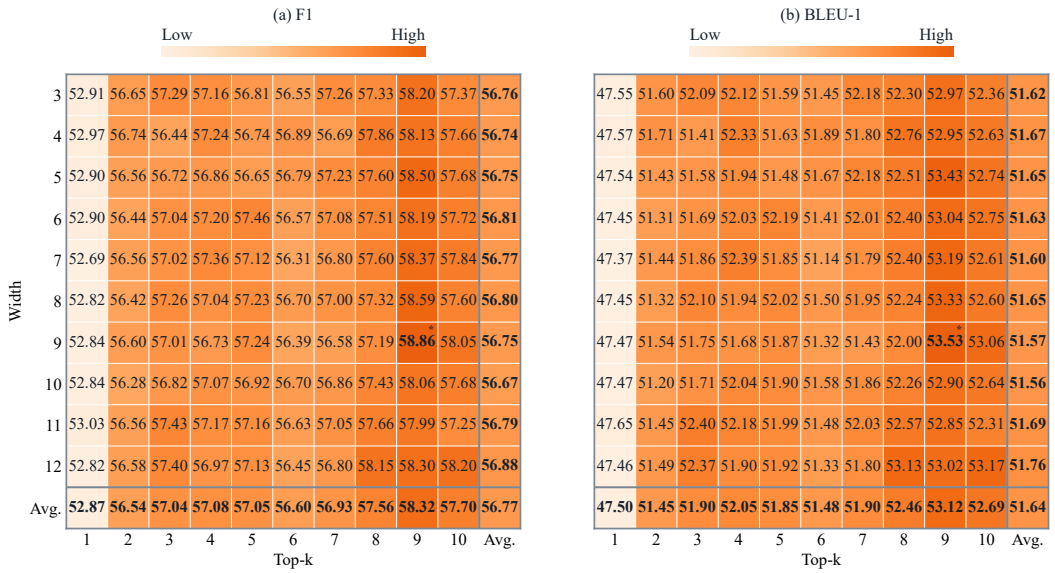

Figure 6: Beam-search hyperparameter robustness, macro-averaged over six LoCoMo conversations. Panels (a) and (b) show F1 and BLEU-1, respectively. The rightmost column and bottom row report averages over Beam Top-k and Beam Width, and the asterisk marks the highest score within each metric. Colors are normalized independently in the two panels.

ercased, punctuation and the English articles *a*, *an*, and *the* are removed, and redundant whitespace is normalized. All reported metrics are scaled to the range of 0–100.

Inference-cost measurement. Figure 4(a) reports six-conversation macro-averaged F1 alongside average question-answering input and output tokens per question. Token-cost records cover 802 questions for Mem0 and 1,540 for RAG, SimpleMem, Mem-α, and MemGPT; sample counts for the remaining methods and the precise cost boundaries for Mem0 and SimpleMem are not fully documented. F1 and cost may therefore use different question sets. The candidate frontier summarizes the displayed observations; it does not establish a controlled efficiency ranking. Panels (b–e) vary the allowed retrieval-round limit and do not report actual rounds or token consumption, so their curves should not be used to infer the costs in panel (a).

Analysis configuration. The Multi-Round Dynamics analysis tracks the checkpoints obtained after Cycles 1–9.

For Organizer–Navigator compatibility, Organizer checkpoints from Cycles 1, 5, and 9 are paired with Navigator checkpoints from Cycles 1, 3, 5, 7, and 9, yielding 3×5 configurations. For a fixed Organizer checkpoint, its constructed memory trees are reused across different Navigator checkpoints so that the memory organization remains unchanged within each row of the compatibility matrix.

For cross-model compatibility, we combine Organizers and Navigators instantiated from Qwen2.5-1.5B (Qwen Team, 2025), Qwen3-4B-Instruct-2507 (Yang et al., 2025), Qwen2.5-7B-Instruct, and LLaMA3.1-8B (Grattafiori et al., 2024), yielding 4×4 backbone combinations. All analyses follow the same LoCoMo evaluation protocol as the main experiments.

A.4 Beam Search Hyperparameter Robustness

We evaluate the robustness of the navigation procedure across Beam Width values from 3 to 12 and Beam Top-k values from 1 to 10. Each configuration is macro-averaged over six LoCoMo conversations. Figure 6 reports the complete sweep for F1 and BLEU-1.

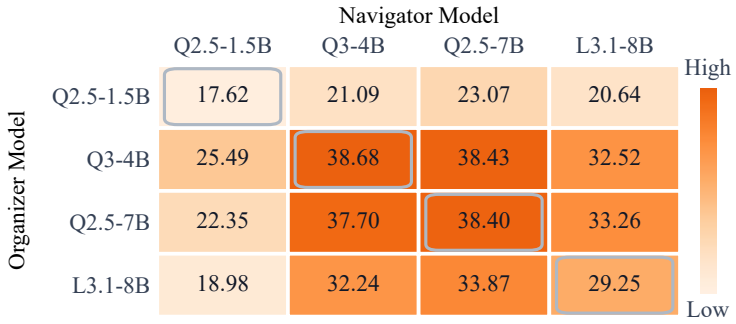

Figure 7: Overall F1 for Organizer–Navigator pairs instantiated from four backbones, without additional training under our framework. Rows denote Organizers and columns denote Navigators.

Performance is highly stable with respect to Beam Width. After averaging over Top-k, the F1 row averages range only from 56.67 to 56.88, while the BLEU-1 row averages range from 51.56 to 51.76; the spans are approximately 0.21 and 0.20 points, respectively. Neither metric exhibits a consistent increasing trend as additional parallel search trajectories are maintained.

Beam Top-k has a larger effect at the most restrictive candidate budget, but otherwise also exhibits a broad stable region. Top-k $= 1$ yields column averages of 52.87 F1 and 47.50 BLEU-1, showing a consistent decline when too few candidate nodes are retained at each navigation step. Across Top-k values from 2 to 10, however, the column averages remain within 56.54–58.32 for F1 and 51.45–53.12 for BLEU-1. Together, these results indicate that the navigation procedure is robust across a wide range of beam settings once the candidate budget is no longer overly restrictive, rather than depending on a narrow hyperparameter region.

A.5 Training and Inference Infrastructure

We implement alternating reinforcement learning with Slime, using Megatron-LM (Shoeybi et al., 2020) for parameter optimization and SGLang (Zheng et al., 2024) for rollout generation. Training runs on a single node with eight NVIDIA H20 GPUs: two GPUs perform parameter updates with a tensor parallelism degree of 2, while the remaining six generate rollouts. We enable sequence parallelism, activation recomputation, and dynamic batching. For evaluation, we deploy models with vLLM (Kwon et al., 2023) through OpenAI-compatible APIs for memory organization and retrieval, using eight single-GPU instances to process requests in parallel.

A.6 Cross-Model Compatibility

We pair Organizers and Navigators from four backbones of different sizes and architectures, without additional training under our framework. Figure 7 shows that performance varies across pairings, and using the same backbone for both roles does not consistently yield the highest F1. Cross-model pairings can also achieve competitive results: a Qwen3-4B Organizer paired with a Qwen2.5-7B Navigator reaches 38.43 F1, close to the 38.68 obtained using Qwen3-4B for both roles.

These observations suggest that an effective match between organization and retrieval is not guaranteed by using the same model. A shared tree structure provides an interface for memory access, but its usefulness as an implicit memory prior depends on whether the Navigator can interpret the organization strategy and whether that strategy supports question-driven retrieval. This motivates bidirectional learning: the Organizer learns from retrieval feedback, while the Navigator learns to use the resulting organization. Such mutual adaptation offers a way to develop useful implicit memory priors and alleviate the lack of memory priors.

Cross-model compatibility further motivates extending this bidirectional learning process to Organizers and Navigators with different backbones. Although the present experiment does not evaluate such training, it suggests the possibility of jointly adapting models of different sizes for the two roles.

Selecting their capacities according to the computational demands of memory organization and retrieval may offer a better trade-off between inference cost and question-answering performance, providing a promising direction for future work.

A.7 Retrieval-Round Sensitivity by Question Category

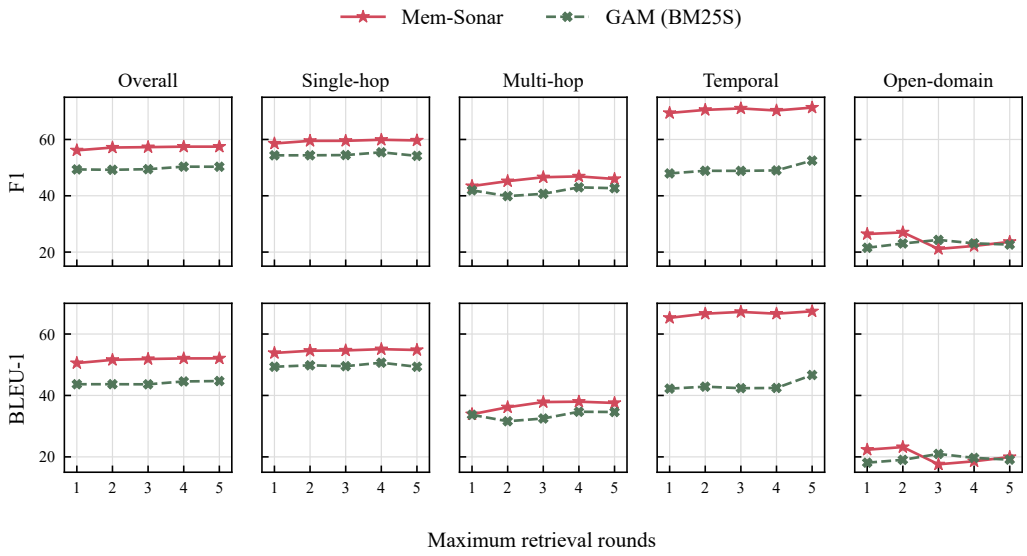

Figure 8: Maximum-retrieval-round sweep for Mem-Sonar and GAM (BM25S) across Overall, Single-hop, Multi-hop, Temporal, and Open-domain categories, evaluated with F1 and BLEU-1.

Figure 8 provides category-level F1 and BLEU-1 results for the Mem-Sonar and GAM comparisons in Figure 4(b–c). Mem-Sonar consistently maintains higher Overall performance across different maximum retrieval rounds. As the retrieval-round limit increases, its performance generally remains stable with modest improvements in several categories, while some categories exhibit non-monotonic fluctuations. In particular, Multi-hop questions benefit more clearly from additional retrieval opportunities before saturating, whereas Open-domain performance is more variable across different round limits. These results suggest that increasing the retrieval-round budget can improve access to relevant evidence, but the benefit depends on the question type and does not grow monotonically with additional rounds.