HyMem: Hybrid Memory Architecture with Dynamic Retrieval Scheduling

Anonymous authors
Paper under double-blind review

Abstract

Large language model (LLM) agents demonstrate strong performance in short-text contexts but often underperform in extended dialogues due to inefficient memory management. Existing approaches face a fundamental trade-off between efficiency and effectiveness: memory compression risks losing critical details required for complex reasoning, while retaining raw text introduces unnecessary computational overhead for simple queries. The crux lies in the limitations of monolithic memory representations and static retrieval mechanisms, which fail to emulate the flexible and proactive memory scheduling capabilities observed in humans, thus struggling to adapt to diverse problem scenarios. Inspired by the principle of cognitive economy, we propose HyMem, a hybrid memory architecture that enables dynamic on-demand scheduling through multi-granular memory representations. HyMem adopts a dual-granular storage scheme paired with a dynamic two-tier retrieval system: a lightweight module constructs summary-level context for efficient response generation, while an LLM-based deep module is selectively activated only for complex queries, augmented by a reflection mechanism for iterative reasoning refinement. Experiments show that HyMem achieves strong performance on both the LOCOMO and LongMemEval benchmarks, outperforming full-context while reducing computational cost by 92.6%, establishing a state-of-the-art balance between efficiency and performance in long-term memory management.

1 Introduction

Large language models (LLMs) inherently operate in a stateless manner, which limits their reasoning capabilities in long-context conversation scenarios Chen et al. (2024); Li et al. (2024); Tseng et al. (2024). Recent research Chhikara et al. (2025); Liu et al. (2024b); Packer et al. (2023); Roucher et al. (2025); Zhong et al. (2024) has primarily focused on designing memory systems to enable efficient storage and retrieval of information for LLM-based agents. These approaches typically treat memory as external knowledge stored in vector databases Edge et al. (2024); Lewis et al. (2020); Shi et al. (2024). To mimic organizational principles of human memory, works such as Chhikara et al. (2025); Li et al. (2025a); Xu et al. (2025); Edge et al. (2024); Han et al. (2024) extract entities and construct relational structures among memory objects. Concurrently, several studies Wang & Chen (2025); Wang et al. (2025); Hu et al. (2025) have proposed multi-category memory segmentation and batch retrieval schemes, thereby providing fundamental support for personalized user memory needs. However, existing methods face two core challenges. First, a single granularity of memory storage struggles to achieve an optimal balance between efficiency and effectiveness. To reduce computational overhead in long dialogues, studies like Fang et al. (2025); Chhikara et al. (2025); Packer et al. (2023); Pan et al. (2024) attempt to compress or distill raw text into lightweight memory libraries. Yet, such compression often leads to loss of critical details, resulting in irreversible "forgetting" phenomena when handling complex or detail-dependent queries Yang et al. (2026); Tan et al. (2025). Conversely, storing raw text minimizes information loss but introduces unnecessary computational burden for simple queries. Second, static memory retrieval strategies fail to effectively emulate the dynamic, hierarchical recall patterns exhibited by humans in complex tasks. These limitations collectively constrain the flexibility and reliability of language agents in long-dialogue applications.

In contrast, the human memory system demonstrates notable cognitive economy: for routine queries, the brain prioritizes low-cost summary retrieval to quickly grasp event outlines; deeper logical reasoning and causal analysis are activated only when tasks require detailed support. This hierarchical,

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.

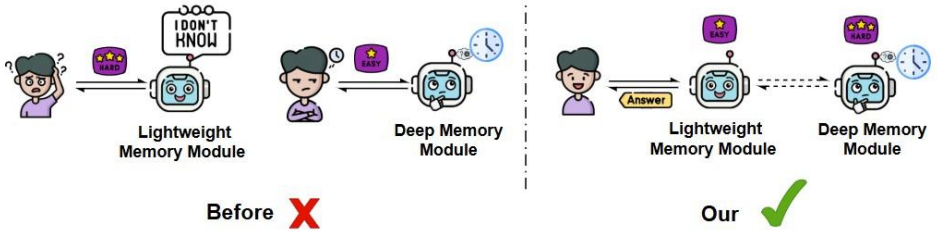


Figure 1: Conventional lightweight methods struggle with complex tasks, while sophisticated approaches incur high overhead for simple queries. In contrast, our HyMem dynamically allocates memory resources based on task demands, achieving dual optimization of performance and efficiency.

adaptive memory scheduling mechanism efficiently integrates memories at different levels, offering valuable inspiration for constructing LLM memory systems that reconcile efficiency with expressive power.

To address these challenges, we propose HyMem, a hybrid memory architecture integrated with a dynamic retrieval mechanism, designed to flexibly adapt to diverse QA scenarios. At the storage level, the architecture partitions raw dialogues into event units and constructs a dual-granularity storage structure comprising summary-level (Level-1 memory) and raw-text-level (Level-2 memory). During inference, HyMem adopts a dynamic retrieval strategy guided by cognitive economy: it first employs lightweight matching to locate Level-1 memory for rapid context construction; then, based on query complexity, it decides whether to activate Level-2 memory for finer-grained information. Within the deep retrieval module, the system performs coarse-grained recall to narrow candidate scope, followed by LLM-powered event-query relevance analysis for precise context localization. Additionally, a reflection module assesses answer completeness and triggers multi-round query refinement when necessary, enhancing the robustness and thoroughness of reasoning. Our main contributions are:

- We propose HyMem, a hybrid memory architecture grounded in the principle of cognitive economy. Through dual-granularity memory storage and dynamic on-demand scheduling, it achieves an effective balance between memory management efficiency and reasoning performance for large-scale language agents in long dialogues.
- We introduce an LLM-guided self-retrieval mechanism that dynamically activates deep memory access, combined with an LLM self-reflection process, enabling precise information localization and efficient, reliable iterative reasoning in complex query scenarios.
- We conduct extensive experiments on the LoCoMo and LongMemEval benchmarks. Results demonstrate that our approach yields significant advantages in answer flexibility and reliability across diverse scenarios.

2 Related Work

2.1 Memory for LLM Agents

To alleviate the inherent limitations of large language models (LLMs) in multi-turn long-context dialogues, a considerable body of recent research has focused on enhancing agents' memory and adaptive evolution during interactions. Techniques such as Munkhdalai et al. (2024); Bulatov et al. (2022); Su et al. (2024); Chen et al. (2023); An et al. (2024); Liu et al. (2023); Xiong et al. (2024); Wei et al. (2025); Wu et al. (2026); Zhang et al. (2025) inject additional signals during training or modify model architectures, aiming to improve retention and reasoning over distant information. However, these approaches generally require retraining the base model, incurring substantial computational and engineering expenses, and are incompatible with powerful closed-source models like GPT-4.

In contrast, retrieval-augmented generation (RAG) methods based on external knowledge bases have become a mainstream, efficient, and flexible solution Hu et al. (2025); Huang et al. (2026); Jin et al. (2025); Wu et al. (2025a); Guo et al. (2024); Lee et al. (2024); Asai et al. (2024). For example, Packer

et al. (2023) adopts a hierarchical memory structure and context paging, leveraging a controller to manage prioritized reading and writing to external vector databases, thereby achieving persistent long-term memory and dynamic injection. Li et al. (2025b) transforms the interaction process into an agent-oriented "memory operating system," which automatically extracts and stores user-relevant information. Approaches such as Li et al. (2025a); Xu et al. (2025); Jimenez Gutierrez et al. (2024); Edge et al. (2024); Xia et al. (2025); Anokhin et al. (2024); Wu et al. (2025b); Rasmussen et al. (2025) draw inspiration from knowledge graphs to organize memories by extracting entities and constructing relationships among memory objects. Meanwhile, Xu et al. (2025) proposes a self-updating paradigm, encompassing memory writing, retrieval, updating, and forgetting. Methods in Wang & Chen (2025); Wang et al. (2025); Hu et al. (2025) introduce multiple structured memory modules and autonomous routing, enabling more fine-grained and personalized long-term memory.

Nevertheless, during the inference phase, prevailing approaches still predominantly rely on fixed memory retrieval workflows—such as those dependent on static vector representations and similarity search—for context construction. These methods exhibit significant limitations when confronted with compositional complex reasoning tasks, presenting a critical bottleneck that restricts agent interaction capabilities in long-context scenarios. To address this issue, our hybrid memory framework, HyMem, adaptively responds to query difficulty, empowering the LLM to proactively retrieve information necessary for complex questions rather than relying solely on vector embeddings. Furthermore, the framework incorporates a reflection mechanism to mitigate hallucination issues arising from unimodal retrieval in challenging scenarios.

2.2 Context Management.

Another important challenge is the rapid increase in token consumption as dialogue context expands. Liu et al. (2026); Fang et al. (2025) introduce token filtering and memory pre-compression to reduce retrieval and generation costs while preserving key information. Xu et al. (2025) further compresses long conversations into discrete "atomic notes," effectively reducing token redundancy. Chhikara et al. (2025) extracts factual knowledge from dialogues and integrates it into user profiles, rather than storing entire conversation logs. Wang et al. (2024) compresses historical information into latent states to form a memory pool. However, such compression inevitably leads to permanent information loss. To address this, Yan et al. (2025); Tan et al. (2025) utilize only raw text to construct context, aiming to avoid information loss induced by memory compression. Recent studies such as Yu et al. (2025); Zhou et al. (2025); Ye et al. (2025) have effectively mitigated the forgetting" problem in long-context reasoning by deferring memory updates to the inference phase, demonstrating notable performance. However, since a full memory update is enforced as a mandatory step in every inference, it inevitably introduces substantial redundant information when processing most simple queries, leading to significant computational overhead and resource waste. Consequently, developing multi-granular memory storage strategies coupled with an adaptive, dynamic scheduling mechanism has become a key challenge to enhancing the efficiency of LLM-based agents. To address this, our proposed hybrid memory framework, HyMem, establishes a two-tier storage architecture consisting of event-level summaries (Level-1 memory) and raw dialogue text (Level-2 memory). By dynamically selecting the appropriate memory granularity based on query complexity, HyMem not only preserves answer accuracy but also significantly improves inference efficiency.

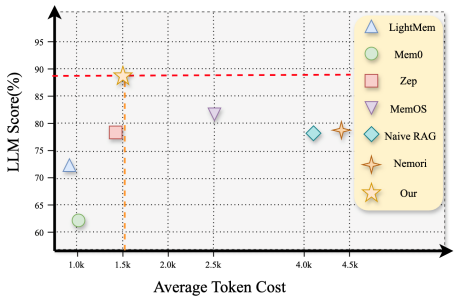


Figure 2: Our approach demonstrates superior efficiency, achieving the best balance between performance and computational cost on the LOCOMO benchmark.

3 Method

We propose HyMem, a memory architecture comprising four core components: a memory storage module, a lightweight memory module, a deep memory module, and a reflection module. The overall design aims to efficiently simulate the multi-level memory processing observed in humans, enabling agents to store and retrieve information flexibly and effectively in long-dialogue scenarios.

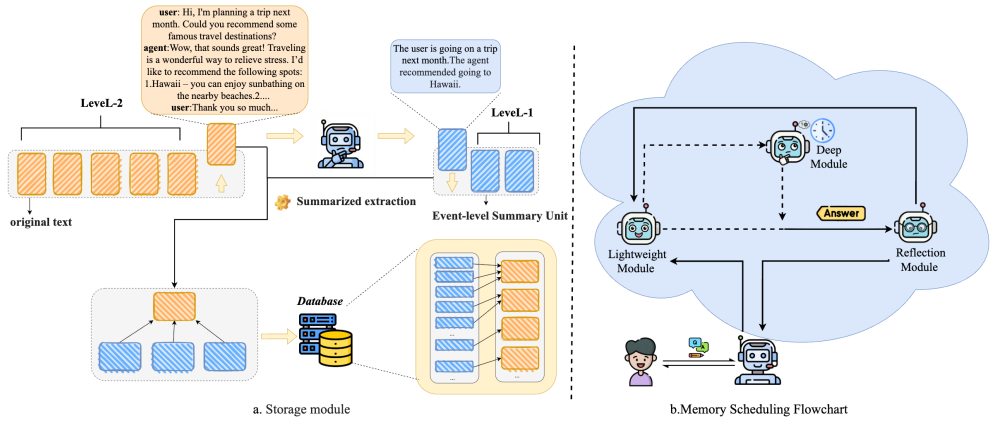


Figure 3: (a) Workflow of the Memory Storage Module. This diagram illustrates the complete pipeline for constructing the dual-layer memory structure: generating core summaries (Level-1 memory) from pre-partitioned event units of raw dialogues (Level-2 memory), establishing many-to-one links to support memory backtracking during the recall stage, before final persistent storage in the database. (b) Inference Workflow within the Memory Recall Module. Queries are first routed to a lightweight module, with selective activation of a deep module when necessary, followed by iterative optimization and final review by the Reflection Module to produce the output.

3.1 Memory Storage Module

Current approaches often treat the majority of content in long dialogues as redundant, arguing that such redundancy not only increases token consumption during inference but also obscures critical information. However, information deemed low-value may not be useless, and over-compression inevitably leads to the loss of useful content—an issue we will analyze in detail in Section 4.5. While we acknowledge the value of distilled memory for efficient storage and question answering—experiments on the LOCOMO benchmark show that in approximately 70% of cases, answers generated using distilled memory combined with a naive RAG strategy suffice—the remaining 30% reveal critical bottlenecks. Among these failures, about 15% and 5% are attributable to retrieval errors and information loss due to compression, respectively. As shown in Figure 3(a), to flexibly accommodate diverse scenario requirements, HyMem's memory storage module first partitions the raw dialogue into discrete event units based on discussed topics. To maintain coherence between events, we allow content overlap across different units. For each abstracted event, core information is extracted. Specifically, the pipeline distills the general progression of the event and key elements that can be directly associated with queries—such as context, time, location, and participants—into a coherent summary, while omitting dialogue details, causal relationships, and social conventions as redundant content. This forms a Level-1 memory unit s_i. The corresponding raw dialogue text for each event is preserved as a Level-2 memory unit p_i.

During the storage phase, the system performs two main operations: First, each Level-1 memory unit is vectorized using an embedding model E , such that $E(s_i) = e_i$, to support efficient retrieval in the lightweight module. Unlike Tan et al. (2025), Level-1 memory units in HyMem serve not only as retrieval entries but also as raw material for context construction, enabling efficient recall and lightweight reasoning throughout the inference process. Second, explicit links are established between each Level-1 memory unit and its corresponding Level-2 memory unit. This design allows the deep memory module, upon detecting a potential logical association between a query and s_i, to backtrack to the detailed content in Level-2 memory, thereby reconstructing a more informative and reliable context. Both types of memory units are stored synchronously in the memory database, providing multi-granular support for retrieval and reasoning.

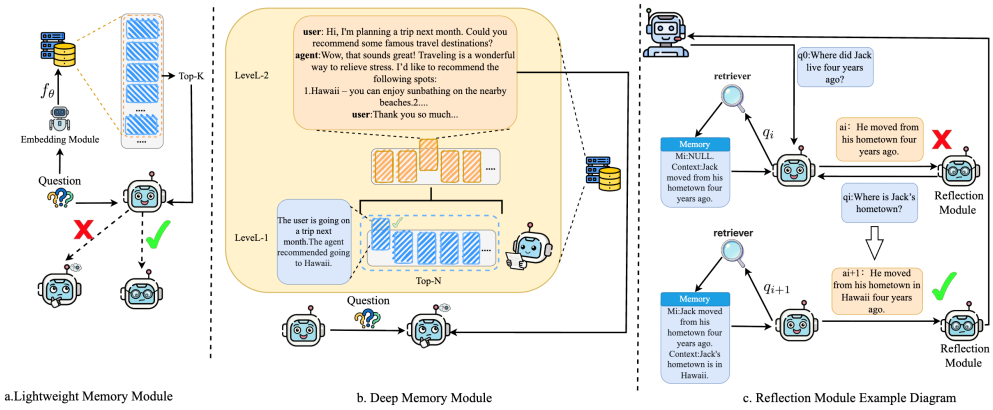


Figure 4: (a) Workflow of the Lightweight Module. (b) Context Reconstruction via the Deep Memory Module by Backtracking from Level-1 to Level-2 Memory. (c) Process of Review and Iterative Optimization by the Reflection Module.

3.2 Light Memory Module

During the inference process, we maintain a memory pool M_i, which is initially empty and used to store intermediate answers generated during the question-answering process. The initial user query is denoted as q_0, and q_i represents the intermediate query at the i -th reasoning step; in the first iteration, $q_i$is initialized to q_0.

As illustrated in Figure3(b), we introduce a dynamic memory scheduling mechanism that adaptively responds to input queries. In accordance with the cognitive economy principle—which mimics energy-efficient human recall—the incoming query q_i is first sent to the lightweight memory module L for vectorization. As shown in Figure4(a), module L retrieves the top-k most relevant segments from the Level-1 memory units using cosine similarity, constructs a lightweight context, and attempts to generate an intermediate answer. This process is characterized by low latency and minimal computational overhead, making it suitable for most simple query scenarios. Additionally, the lightweight memory module evaluates the contextual state to determine whether to invoke the deep memory module. If it detects contextual incompleteness—defined as a "forgetting" state due to either retrieval failure or information loss—the deep memory module is dynamically activated, and q_i is forwarded for more precise retrieval and Level-2 memory reconstruction. If the context is deemed complete, the answer is sent directly to the reflection module for further evaluation, and the memory pool M_i is updated accordingly.

3.3 Deep Memory Module

As illustrated in Figure 4(b), the deep memory module consists of a retriever f_γ and a generator G_γ. Unlike the lightweight module, which relies solely on semantic similarity for retrieval, this module adopts a two-stage retrieval strategy: first, it performs coarse-grained recall based on semantic similarity to select the top-N candidate events, narrowing the search scope; subsequently, it introduces an LLM-based self-retrieval mechanism to deeply identify key events s_k in the Level-1 memory that exhibit semantic relevance, logical connection, or causal dependency with the query q_i. The retriever then backtracks to the corresponding Level-2 memory unit p_k via pre-established links, reconstructing a context with comprehensive information. This context, along with the original query q_0, is fed into the generator G_γ to produce the final answer, and the memory pool M_i is updated accordingly.

HyMem defers reasoning about inter-event relationships and query decomposition to the reflection stage. During a single retrieval process, each event is treated as an independent, atomic memory unit. Thanks to the decoupled architecture of the retriever and generator, the retriever can process different batches of Level-1 memory units in parallel, without considering dependencies among units.

Specifically, the candidate set obtained from coarse recall can be partitioned by a batch size d :

$$S = \bigcup_{i=1}^{m/d} S_i, \quad S_i = \{s_{(i-1)d+1},\ s_{(i-1)d+2},\ \ldots,\ s_{id}\} \tag{1}$$

. Our approach effectively avoids the $O(N)$ time complexity inherent in traditional streaming memory processing schemesYu et al. (2025). The overall inference complexity of the system primarily depends on the number of iteration rounds in the reflection phase and the concurrency of LLM calls, thereby significantly improving processing efficiency while maintaining retrieval quality.

3.4 Reflection Module

Before the system outputs the final answer, the reflection module R conducts a comprehensive completeness check. Unlike the lightweight contextual completeness assessment performed by the memory module, the reflection module autonomously plans subsequent reasoning steps based on the response completeness. Specifically, this module rigorously evaluates whether the query has been decomposed into minimal answerable units and whether each unit has been correctly addressed. If further responses are possible, the reflection module decomposes the original problem into sub-problems and reformulates the query, generating q_{i+1} for the next round of retrieval. If the problem is fully resolved, the process terminates and the final answer is produced.

This mechanism is particularly critical for complex queries, whose answers may depend on multiple memory segments. As illustrated in Figure 4(c), although memory units may be independent, the chained combination of queries can give rise to progressive causal or dependency relationships. Since each retrieval round is treated as an atomic operation, a single retrieval may only yield a superficial answer, while underlying dependencies could lead to incomplete responses. The reflection and query rewriting process effectively mitigates this limitation, substantially enhancing the system's ability to handle complex queries.

4 Experiment

Our experiments consist of six parts: (1) as shown in Sections 4.1 and A.1, we conduct detailed comparisons between HyMem and representative baselines on LOCOMO and LongMemEval; (2) in Section 4.2, we analyze the efficiency–effectiveness trade-off of HyMem in terms of token usage and inference latency; (3) we compare HyMem with Naive RAG under different budgets, considering both performance and computational cost; (4) we perform ablations to quantify the contributions of Level-1/Level-2 granularity, the lightweight/deep memory modules, and the reflection module; (5) we study information loss induced by memory pre-compression and validate our event-level compression strategy; and (6) we sweep the retrieval hyperparameter k in the lightweight memory module.

4.1 Main Result

On the LOCOMO dataset, we set the retrieval count k of the lightweight module to 10 and the coarse retrieval count N of the deep memory module to 30 for experimentation. As summarized in Table 1, our method demonstrates competitive performance on the single-hop task (ranking second), while consistently exceeding all baseline models across all other task categories as well as in overall average accuracy. Notably, HyMem achieves particularly strong results on multi-hop and open-domain tasks, with accuracies of 88.16% and 77.08%, respectively. Even when compared to baseline methods with full context access, it yields substantial improvements of 10.46% and 5.20% points, clearly validating the efficacy of the proposed deep memory module in handling complex reasoning challenges. Furthermore, on the LongMemEval-S benchmark, our approach attains a leading score of 78.80%, surpassing all baseline methods. These results indicate that our method exhibits strong competitive advantages in scenarios involving ultra-long contexts.

4.2 Efficiency Analysis

On the LoCoMo dataset with the gpt-4o-mini backbone, we compare HyMem with other mainstream baselines in terms of average per-sample inference latency and average token consumption per

Table 1: Comparison of LLM-as-a-Judge scores (%) between HyMem and each baseline on the LOCOMO dataset and LongMemEval-s.

	Method	LOCOMO						LongMemEval-s
		Avg Tokens	Single Hop	Multi Hop	Open Domain	Temporal	Overall	Overall
gpt-4o-mini	A-Mem	2.7k	40.22	19.01	55.10	50.12	49.01	62.60
	Mem0	1.7k	67.13	51.15	72.93	55.51	66.88	53.61
	Zep	1.4k	74.11	66.04	67.71	79.76	75.14	61.20
	LightMem	0.8k	76.61	67.02	45.83	76.32	72.99	68.64
	Nemori	4.7k	82.10	65.30	44.80	71.00	74.4	64.20
	Full-Context	21.4k	80.14	46.89	58.33	90.49	77.51	56.80
	Our	1.2k	72.34	80.06	71.88	88.82	82.92	75.00
gpt-4.1-mini	Zep	1.4k	79.43	69.16	73.96	83.33	79.09	71.12
	Mem0	1.0k	62.41	57.32	44.79	66.47	62.47	66.40
	Mirix	5k	85.11	83.70	65.62	88.39	85.38	56.80
	Nemori	4.4k	84.90	75.10	51.00	77.60	79.40	74.60
	MemU	4.0k	74.91	72.34	43.61	54.17	66.67	38.40
	Naive RAG(k=3)	2.5k	70.92	69.47	65.62	75.74	72.97	67.20
	Naive RAG(k=5)	4.1k	72.69	74.14	68.75	81.21	77.40	72.80
	Full-Context	21.4k	**88.53**	77.70	71.88	92.70	87.52	58.20
	Our	1.5k	85.15	**88.16**	**77.08**	**92.98**	**89.55**	**78.80**

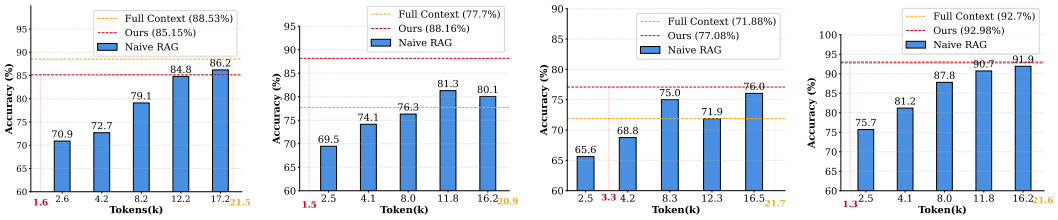
question. As shown in Table 2, HyMem achieves the best score and significantly outperforms other methods in average per-sample inference efficiency; relative to Mem0 and SimpleMem, HyMem reduces retrieval-stage latency by 83.3% and 74.9%, respectively. We attribute these gains to HyMem's dynamic scheduling mechanism, which further decreases inference latency. In most query scenarios, the system relies only on lightweight modules for memory recall and context construction, and selectively activates the deep-memory module only for complex questions, thereby avoiding the high average latency induced by always using complex retrieval strategies. These results validate the effectiveness of HyMem's dynamic memory scheduling strategy. By adaptively allocating computational resources according to query complexity, HyMem substantially reduces average computation cost while achieving an optimal balance between performance and efficiency.

Table 2: Inference latency and token consumption comparison on the LOCOMO dataset.

Metric	A-Mem	LightMem	Mem0	SimpleMem	REMem	RF-mem	Mem1	HyMem
Inference Time (s)	796.7	577.1	583.4	388.3	312.0	497.0	283.9	**97.4**
Cost Tokens (k)	2.7	0.8	1.7	5.4	9.4	9.4	9.6	1.2
LLM Score	49.01	72.99	66.88	71.43	74.02	76.49	72.60	**82.92**

4.3 Comparison with a Naive RAG

We compare the efficiency and effectiveness of Naive RAG approaches with our proposed method under varying numbers of retrieved items k . The experiments employ text-embedding-3-small as the embedding model and are evaluated on four types of tasks from the LOCOMO dataset. For Naive RAG, we test accuracy and inference token consumption for $k \in \{3, 5, 10, 15, 20\}$. To maximize baseline performance, retrieval is performed directly on the original memory segments rather than compressed representations. Our method uses the same embedding model but fixes $k = 10$ in the retrieval module. As shown in Figure 5, the accuracy of Naive RAG initially increases with larger k , which is expected since a higher recall count improves the likelihood of retrieving relevant events. However, beyond a certain threshold, performance plateaus or even declines. This observation aligns with findings from Fang et al. (2025); Liu et al. (2024a); Pan et al. (2025), and our experiments confirm that excessively redundant context leads to diminishing returns. In contrast, HyMem achieves significantly lower inference token overhead while attaining higher accuracy than Naive RAG under equivalent cost settings—even outperforming the full-context baseline.



(a) Single Hop tasks. (b) Multi Hop tasks. (c) Open Domain tasks. (d) Temporal tasks.

Figure 5: Comparison of performance and average token usage on the four LOCOMO task categories for Naive RAG with different retrieval k values, Full Context, and our method.

4.4 Ablation Studies

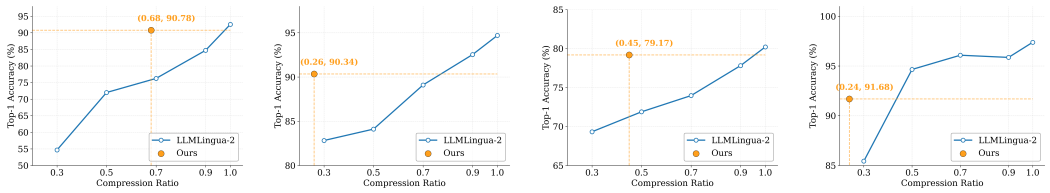
We conduct a comprehensive ablation study on the LOCOMO dataset to evaluate the impact of memory granularity in HyMem's storage module—core summaries as Level-1 memory versus raw text as Level-2 memory—along with the three inference-time modules: the lightweight memory module, deep memory module, and reflection module. Table 3 summarizes the experimental results. Notably, removing both the deep memory module and Level-2 memory considerably reduces inference token usage (by 700–800 tokens), but at the cost of a 17.40% drop in accuracy, which we attribute to the coarsening of information granularity and suboptimal retrieval performance. In contrast, removing the lightweight memory module yields a slight accuracy improvement of 2.20%; however, this forces all queries to be processed exclusively by the deep memory module, increasing the average token consumption to 3192. Since approximately 70% of queries can be handled efficiently by the lightweight module in practice, omitting this component leads to substantial computational inefficiency. Furthermore, ablating the reflection module results in accuracy reductions of 2.21%, 4.17%, 0.23%, and 1.09% under different configurations, highlighting its crucial role in enhancing the agent's ability to manage complex multi-turn dialogues—a process that also incurs a modest rise in token usage. Lastly, utilizing Level-2 memory in place of Level-1 memory improves accuracy by about 6.34%, albeit at the expense of increasing average token consumption by roughly fivefold.

4.5 Compression Loss Analysis

We hypothesize that the errors made by agents in long-context tasks mainly stem from the following factors: (1) Retrieval Errors: Retrieved passages are irrelevant to the query or incomplete in content; (2) Over-compression: Information loss caused by aggressive compression during memory storage, leading to insufficient evidence retrieved for reasoning; (3) Model Reasoning Errors: The model still outputs incorrect answers even when provided with the complete context. This is often due to excessive textual redundancy, which hinders the formation of effective context associations, or due to the inherent complexity of the query exceeding the model's reasoning capabilities; (4) Dataset Errors: The dataset itself contains intrinsic flaws. Since the last two types of errors (Model and Dataset Errors) are inherent to the model and dataset, we isolate and evaluate the impact of compression granularity solely by removing the retrieval module and directly providing the compressed 'ground-truth' passages as context to the model. Specifically, we employ the LLMLingua-2 model to pre-compress the original text and evaluate the performance across the four task types in the LOCOMO benchmark under varying compression ratios, $ratio \in \{0.3, 0.5, 0.7, 0.9, 1.0\}$. The setting with a compression ratio of 1.0 (the full, uncompressed passage is provided) serves as an approximate upper bound of the model's performance on LOCOMO.

Storage		**Retrieval**			**Overall**	**Token**
Level-1	Level-2	L	D	R		
✓	✓	✓	✓	✓	89.55	1540
✓		✓		✓	72.15	850
✓	✓		✓	✓	91.75	3192
✓	✓	✓	✓		87.34	1411
✓		✓			67.98	794
✓	✓		✓		91.16	2947
	✓	✓		✓	78.49	4125
	✓	✓			77.40	4046

Table 3: Impact of Memory Granularity (Core Summary Level-1, Original Text Level-2) and Retrieval Modules (Lightweight, Deep, Reflection) on Overall (%) and Avg Token Consumption.



(a) Single Hop tasks. (b) Multi Hop tasks. (c) Open Domain tasks. (d) Temporal tasks.

Figure 6: Evaluating accuracy loss under different compression ratios using the LLMLingua-2 pre-compression method.

As shown in Figure 6, after eliminating noise introduced by retrieval, task accuracy declines as the compression ratio decreases, validating our initial hypothesis that fine-grained memory is critical for certain QA tasks. Further experiments demonstrate that our proposed event-level compression method, which reduces memory length to approximately 30% of the original text, significantly outperforms LLMLingua-2 in accuracy—even under the same or more aggressive compression ratios. This indicates that our compression strategy more effectively preserves information crucial for downstream reasoning while efficiently controlling textual redundancy.

4.6 Hyperparameter Analysis

As shown in Table 4, increasing the lightweight module's recall count k produces only modest accuracy changes (the reported variance is approximately 0.60), because HyMem activates the deep module whenever lightweight retrieval is insufficient. Larger k improves lightweight recall and therefore reduces costly deep-module invocations, lowering total token use for $k < 10$. Once $k > 10$, the invocation rate saturates; further increases yield diminishing accuracy gains while adding token overhead from the expanded lightweight context.

Table 4: Effect of recall count k on accuracy, token use, and deep-module invocation rate.

k	1	3	5	7	10	12	15	20
Overall	87.40	88.25	88.63	88.89	89.55	89.63	89.57	89.74
Token	1966	1747	1561	1558	1540	1594	1647	1729
Deep ratio	0.43	0.32	0.29	0.27	0.22	0.22	0.22	0.21

5 Conclusion

We propose HyMem, a hybrid memory framework for large language model agents inspired by the principle of cognitive economy, designed to address the efficiency-performance trade-off in long dialogue scenarios. Unlike traditional static memory approaches, HyMem employs a dual-granularity memory storage and dynamic scheduling mechanism that strategically allocates retrieval and reasoning resources between a lightweight memory module and an activatable deep memory module based on query complexity. Augmented by a reflection module for multi-turn iterative refinement, our framework significantly enhances the flexibility and robustness of memory management. Experimental results show that HyMem achieves state-of-the-art performance across multiple widely used benchmarks. This work underscores the importance of adaptive, resource-aware memory systems in enabling long-term, robust, and efficient agent interactions, and lays a foundation for theoretical and practical advances in scalable dialogue agents.

AI use statement

Generative AI was used to assist with migrating this manuscript into the ICLR 2027 LaTeX template and checking its formatting. It was not used during this migration to generate experimental results or alter the scientific claims. The authors reviewed the migrated manuscript and take responsibility for

its final content. The authors should update this statement before submission if any additional uses of generative AI must be disclosed under the ICLR 2027 policy.

Reproducibility statement

The main paper reports the experimental settings, evaluation protocol, efficiency analysis, ablations, and hyperparameter analysis. The appendix provides additional benchmark results, dataset and baseline descriptions, algorithmic details, and the complete system prompts used in the evaluation.

References

Chenxin An, Fei Huang, Jun Zhang, Shansan Gong, Xipeng Qiu, Chang Zhou, and Lingpeng Kong. Training-free long-context scaling of large language models. *arXiv preprint arXiv:2402.17463*, 2024.

Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. Arigraph: Learning knowledge graph world models with episodic memory for llm agents. *arXiv preprint arXiv:2407.04363*, 2024.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *International Conference on Learning Representations*, 2024.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, et al. When large language models meet personalization: Perspectives of challenges and opportunities. *World Wide Web*, 27(4):42, 2024.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023.

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

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

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

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*, 2024.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*, 2024.

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

Wei-Chieh Huang, Weizhi Zhang, Yueqing Liang, Yuanchen Bei, Yankai Chen, Tao Feng, Xinyu Pan, Zhen Tan, Yu Wang, Tianxin Wei, et al. Rethinking memory mechanisms of foundation agents in the second half. *arXiv preprint arXiv:2602.06052*, 2026.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J Taylor, and Dan Roth. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. *arXiv preprint arXiv:2504.14225*, 2025.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569, 2024.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 737–740, 2025.

Myeonghwa Lee, Seonho An, and Min-Soo Kim. Planrag: A plan-then-retrieval augmented generation for generative large language models as decision makers. *arXiv preprint arXiv:2406.12430*, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Rui Li, Zeyu Zhang, Xiaohe Bo, Zihang Tian, Xu Chen, Quanyu Dai, Zhenhua Dong, and Ruiming Tang. Cam: A constructivist view of agentic memory for llm-based reading comprehension. *arXiv preprint arXiv:2510.05520*, 2025a.

Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

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

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024a.

Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*, 2023.

Zhiwei Liu, Weiran Yao, Jianguo Zhang, Liangwei Yang, Zuxin Liu, Juntao Tan, Prafulla K Choubey, Tian Lan, Jason Wu, Huan Wang, et al. Agentlite: A lightweight library for building and advancing task-oriented llm agent system. *arXiv preprint arXiv:2402.15538*, 2024b.

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

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 101, 2024.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. Nemori: Self-organizing agent memory inspired by cognitive science. *arXiv preprint arXiv:2508.03341*, 2025.

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

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*, 2024.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H Vicky Zhao, Lili Qiu, et al. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations*, 2025.

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*, 2025.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. smolagents': A smol library to build great agentic systems, 2025. *URL https://github.com/huggingface/smolagents. GitHub repository*, 2025.

Zeru Shi, Kai Mei, Mingyu Jin, Yongye Su, Chaoji Zuo, Wenyue Hua, Wujiang Xu, Yujie Ren, Zirui Liu, Mengnan Du, et al. From commands to prompts: Llm-based semantic file system for aios. *arXiv preprint arXiv:2410.11843*, 2024.

Yiheng Shu, Saisri Padmaja Jonnalagedda, Xiang Gao, Bernal Jiménez Gutiérrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. Remem: Reasoning with episodic memory in language agent. *arXiv preprint arXiv:2602.13530*, 2026.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, et al. 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, 2025.

Yu-Min Tseng, Yu-Chao Huang, Teng-Yun Hsiao, Wei-Lin Chen, Chao-Wei Huang, Yu Meng, and Yun-Nung Chen. Two tales of persona in llms: A survey of role-playing and personalization. *arXiv preprint arXiv:2406.01171*, 2024.

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

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, et al. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*, 2024.

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

Rubin Wei, Jiaqi Cao, Jiarui Wang, Jushi Kai, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. Mlp memory: A retriever-pretrained memory for large language models. *arXiv preprint arXiv:2508.01832*, 2025.

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

Ruofan Wu, Youngwon Lee, Fan Shu, Danmei Xu, Seung-won Hwang, Zhewei Yao, Yuxiong He, and Feng Yan. Composerag: A modular and composable rag for corpus-grounded multi-hop question answering. *arXiv preprint arXiv:2506.00232*, 2025a.

Yaxiong Wu, Yongyue Zhang, Sheng Liang, and Yong Liu. Sgmem: Sentence graph memory for long-term conversational agents. *arXiv preprint arXiv:2509.21212*, 2025b.

Zijun Wu, Yongchang Hao, and Lili Mou. Tokmem: One-token procedural memory for large language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Siyu Xia, Zekun Xu, Jiajun Chai, Wentian Fan, Yan Song, Xiaohan Wang, Guojun Yin, Wei Lin, Haifeng Zhang, and Jun Wang. From experience to strategy: Empowering llm agents with trainable graph memory. *arXiv preprint arXiv:2511.07800*, 2025.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4643–4663, 2024.

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

BY Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General agentic memory via deep research. *arXiv preprint arXiv:2511.18423*, 2025.

Chengyuan Yang, Zequn Sun, Wei Wei, and Wei Hu. Beyond static summarization: Proactive memory extraction for llm agents. *arXiv preprint arXiv:2601.04463*, 2026.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, et al. Agentfold: Long-horizon web agents with proactive context management. *arXiv preprint arXiv:2510.24699*, 2025.

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

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025.

Yingyi Zhang, Junyi Li, Wenlin Zhang, Penyue Jia, Xianneng Li, Yichao Wang, Derong Xu, Yi Wen, Huifeng Guo, Yong Liu, et al. Evoking user memory: Personalizing llm via recollection-familiarity adaptive retrieval. *arXiv preprint arXiv:2603.09250*, 2026.

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

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

A Additional Results

A.1 Evaluation on Additional Datasets

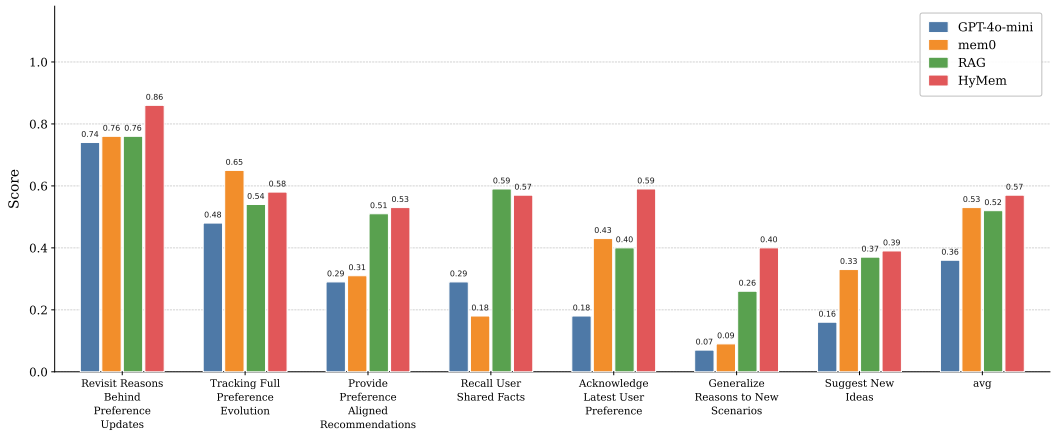


Figure 7: Comparison of performance between mainstream baselines and our method across seven PersonMem-128k task categories.

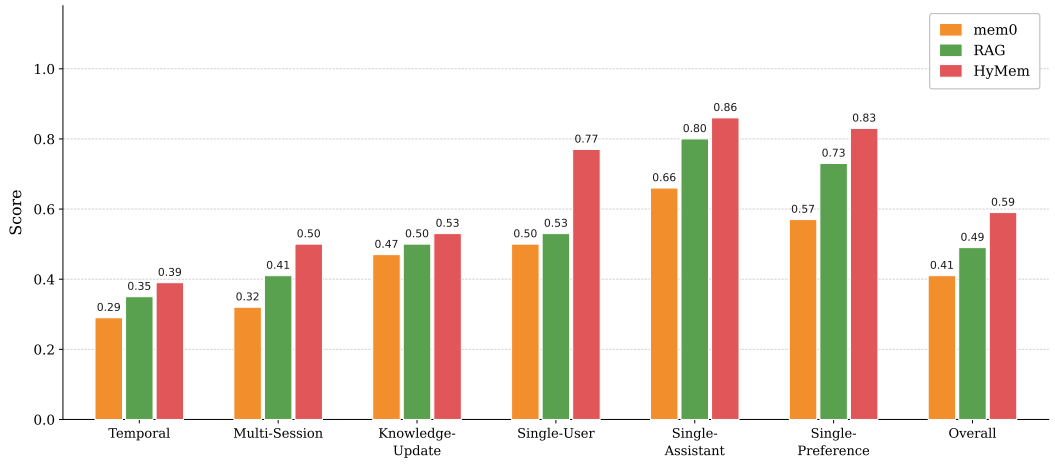


Figure 8: Comparison of performance between mainstream baselines and our method across six LongMemEval-M task categories.

We further evaluate on the more challenging PersonMem-128k and LongMemEval-M benchmarks. As shown in Figure 7 and Figure 8, HyMem outperforms all baselines, further demonstrating its strong applicability even under ultra-long context settings.

A.2 Comparison of the Lightweight and Deep Modules

We ablate the reflection module and separately evaluate the impact of the lightweight module and the deep module on both efficiency and effectiveness, and further compare them with the hybrid setting.

As shown in Table 5, under the same recall budget, complex retrieval improves accuracy by 18.96% over simple retrieval, indicating that complex retrieval serves as a reliable fallback when simple retrieval fails. However, it introduces an additional 40–90 seconds of latency, due to extra token consumption when processing coarse retrieval outputs. As the coarse retrieval scope expands, accuracy increases slightly, but at the cost of higher token usage and latency. Overall, the hybrid

Table 5: Effect of retrieval budget (k) on performance and efficiency for Light, Deep, and Hybrid settings. Total Tokens = Retrieve Tokens + Answer Tokens.

Module	Retrieve k	LLM Score	Retrieve Tokens	Answer Tokens	Total Tokens	Time (s)
Light	10	67.98	0	794	794	76.8
Deep	10	86.94	596	1583	2179	116.8
	20	89.68	941	1633	2574	123.8
	30	91.16	1286	1761	3047	147.8
	50	91.62	2094	1973	4067	170.3
Hybrid (Light, Deep)	(10, 30)	87.34	179	860	1039	94.6

mode (HyMem) attains accuracy close to the best performance of the deep module, while using only one-third of its token budget and reducing latency by 36%, which further validates the efficiency advantage of dynamic scheduling.

A.3 Error Analysis

As shown in Table 6, we run each experiment three times on the LOCOMO dataset using the gpt-4.1-mini setup, and report the mean and standard deviation to quantify run-to-run stability.

Table 6: Error analysis on LOCOMO.

Single-hop	Multi-hop	Open-domain	Temporal	Overall
85.15	88.16	77.08	92.98	89.55
86.17	85.04	78.12	91.79	88.50
84.04	85.35	76.04	91.43	87.86
85.12 ± 1.07	86.18 ± 1.72	77.08 ± 1.04	92.07 ± 0.81	88.64 ± 0.85

B Dataset

B.1 LOCOMO

The LoCoMo datasetMaharana et al. (2024) comprises dialogues with an average total length of 9,000 tokens, where each dialogue contains approximately 600 turns and 26,000 tokens. Each dialogue includes around 200 questions, which are categorized into the following types: (1) single-hop questions that can be answered using information from a single turn; (2) multi-hop questions requiring information integration across different turns; (3) temporal reasoning questions designed to assess a model's ability to reason about sequential events; and (4) open-domain knowledge questions that require combining dialogue context with external knowledge. LoCoMo is particularly suited for evaluating a model's capacity to handle long-range dependencies and maintain consistency throughout extended interactions.

B.2 LongMemEval

The LongMemEval datasetWu et al. (2024) focuses on assessing long-context memory capabilities. It includes high-difficulty tasks such as long-text comprehension, multi-stage reasoning, and information integration across segments. The dataset emphasizes model performance in handling ultra-long contexts, complex causal chains, and long-term information retention. Through diverse task settings, LongMemEval enables a more comprehensive evaluation of memory systems with respect to long-range dependencies, detailed tracking, and information retrieval accuracy and robustness, thus providing more stringent standards for practical model application.

B.3 PERSONMEM

The PersonMem dataset Jiang et al. (2025) focuses on evaluating the personalized response capability of LLMs in long-term user-LLM interactions. It contains over 180 simulated user-LLM interaction histories, each comprising up to 60 multi-turn sessions with a total context length of up to 1 million tokens. Each session consists of 15–30 conversation turns across 15 real-world tasks that require personalization, such as travel planning, therapy consultation, and food recommendation. The dataset assesses a model's ability to (1) internalize user traits and preferences from interaction history, (2) track the dynamic evolution of user profiles over time, and (3) generate responses that align with the user's current state in new scenarios. It includes seven types of in-situ user queries (e.g., recalling user-shared facts, acknowledging latest preferences, tracking full preference evolution, and generalizing to new tasks).

C BASELINES

A-Mem Xu et al. (2025): A memory system that constructs Zettelkasten-style knowledge graphs from user–agent interactions, dynamically associating notes through embedding similarity analysis and LLM-based reasoning.

ZepRasmussen et al. (2025): A commercial memory API that builds temporal knowledge graphs (Graphiti) from user dialogues and metadata, designed for efficient semantic querying.

Mem0 Chhikara et al. (2025): An open-source memory system that incrementally compresses and stores memory facts using LLM-based summarization, with an optional graph memory extension.

REMemShu et al. (2026):A two-phase episodic memory framework that explicitly models experiences as time-aware events. It indexes interactions into a hybrid memory graph that links timestamped natural-language gists with time-scoped factual triples, then performs iterative, tool-augmented agentic inference" (retrieval + graph exploration) to support episodic recollection and multi-step temporal reasoning.

RF-Mem Zhang et al. (2026): An uncertainty-guided dual-path memory retriever for personalized LLMs inspired by the cognitive dual-process theory of *familiarity* vs. *recollection*. It performs a probe retrieval and uses the mean similarity score and entropy as a familiarity signal to adaptively switch between (i) fast one-shot top-K retrieval when confidence is high, and (ii) a multi-round recollection path that clusters candidate memories and iteratively expands evidence via query–centroid α-mixing in embedding space to reconstruct relevant context.

LightMemFang et al. (2025): A lightweight memory system inspired by human memory models, which employs a three-stage mechanism—sensory memory filtering, topic-aware short-term memory organization, and long-term memory updated during sleep time—to significantly improve memory processing efficiency while maintaining performance.

NemoriNan et al. (2025): A self-organizing memory architecture inspired by event segmentation theory and the free-energy principle. It autonomously structures conversational streams into semantically coherent episodes via a Two-Step Alignment principle, and enables agents to proactively learn from prediction errors through a Predict–Calibrate principle, facilitating adaptive knowledge evolution.

MIRIXWang & Chen (2025): A modular, multi-agent memory system that integrates six specialized components—Core, Episodic, Semantic, Procedural, Resource Memory, and Knowledge Vault—coordinated via a multi-agent framework for dynamic memory updates and retrieval. It is particularly adept at handling large-scale multimodal inputs to enable deep personalization.

Full Context: Provide the LLM with all the context corresponding to the question.

Naive RAG: Vectorize the content using the qwen3-0.6b-embedding model and store the entire context as memory for subsequent responses.

D LIMITATIONS

HyMem achieves encouraging results across a broad range of widely used benchmarks, striking a strong balance between effectiveness and efficiency. Nevertheless, several directions remain for

future investigation. First, while HyMem can be readily integrated with both open-source and proprietary foundation models, its ability to plan retrospection paths and identify key events is inherently constrained by the underlying language model's capabilities; consequently, different LLM backbones may yield variations in generation quality. Second, HyMem currently focuses on text-based interactions. Extending the framework to multimodal inputs (e.g., images and audio) is a promising avenue for future work, as it may enable richer context representations.

E ALGORITHM

Algorithm 1 HyMem Memory Retrieval and Reasoning

Input: Initial query q_0, maximum number of iterations T
Initialize: Memory pool $M_0 \leftarrow$ NULL, iteration counter $i \leftarrow 0$
Modules: Light Memory Module $\mathcal{L} = \{$Embedding E, Retriever f_θ, Generator $G_\theta\}$,
Deep Memory Module $\mathcal{D} = \{$ Retriever f_θ, Retriever f_γ, Generator $G_\gamma\}$, Reflection module R
while not `done` and $i < T$ **do**
  $e_{q_i} \leftarrow E(q_i)$ // Embed query
  $[s]_k \leftarrow$ Retrieve top-k Level-1 memories using $f_\theta(e_{q_i})$
  $answer_i, status \leftarrow G_\theta(q_0, [s]_k \cup M_i)$
  if status == `False` **then**
    $[s]_N \leftarrow$ Retrieve top-N Level-1 memories using $f_\theta(e_{q_i})$
    $[s]_m \leftarrow f_\gamma(q_i, [s]_N)$ // LLM retrieval
    $[p]_n \leftarrow$ Map $[s]_m$ to Level-2 memories
    $answer_i \leftarrow G_\gamma(q_0, [p]_n \cup M_i)$
  end if
  Update memory pool: $M_{i+1} \leftarrow answer_i$
  q_{i+1}, `done` $\leftarrow R(answer_i, q_0)$
  if `done` **then**
    Output: $answer_i$
  end if
  $i \leftarrow i + 1$
  $q_i \leftarrow q_{i+1}$
end while

F Detailed System Prompts

In this section, we provide a detailed overview of the prompts employed throughout the HyMem memory system. This includes the summary module in the storage phase, the generator G_θ in the Light Memory Module, the retriever f_γ and generator G_γ in the Deep Memory Module, the reflection module R , as well as the final model used for answer evaluation.

F.1 Summary Module

Your task is to extract all key information from the conversation and summarize each piece into a concise sentence. These sentences will serve as the memory content for subsequent agent responses.
The final output format should be:
{ "keywords": ["Key information 1", "Key information 2", "Key information 3", ...] }
Important instructions:
All key information in the original conversation must be retained, including details such as time, location, persons involved, and significant events, to prevent errors in future responses due to missing details.
Key information should appropriately summarize the valuable content present in the original dialogue. To improve efficiency, all unnecessary dialogue elements (such as greetings, pleasantries, small talk, or casual remarks) must be excluded from the key information.
Summaries should be concise and minimize character usage wherever possible. Whenever possible, integrate multiple related pieces of information into a single key sentence to increase information density. Avoid fragmenting key information excessively, as this could lead to unnecessary character usage.

F.2 Reflection R

Your primary responsibility is to evaluate whether the current answer meets the standard based on the given question and the model's response.
If the answer is irrelevant to the question or contradicts the intent of the question, it should be judged as not meeting the standard. In such cases, set the finished field to 0, and rewrite the question by strengthening it based on what is missing in the answer, so that it can be used for further retrieval.
The newly generated question should be output in the new_question field. If the answer is generally complete and well-reasoned, judge it as meeting the requirements and set the finished field to 1.
Here is an example output: { "finished": 0, "new_question": "..." }

F.3 Generator G_θ

You are a memory-based question answering assistant. You will receive a question along with a summary of related memory.
Your main responsibilities are as follows:
Based on the provided memory summary, determine whether the current question can be answered. If the memory summary does not match the question, is incomplete, vague or ambiguous, is irrelevant, or if you are unsure whether you can answer, you must treat it as unanswerable. Note: Use strict criteria to avoid hallucinations and incorrect responses. In these cases, set the "finished" field to 2. More precise retrieval methods will be used later to provide more complete memory.
If a standard or golden answer to the question is clearly present in the memory summary, generate the answer in the "answer" field and set "finished" to 0. The answer must be complete and specific. For example, for time-related questions, to avoid ambiguity, clearly specify the reference point of any relative time expressions. If the answer is "last year", the correct format should be: "The current year is 2022, so the answer is last year."
The required output format is: {"finished":0, "answer": "..."}
Note: Return output only in JSON format. Do not provide any other form of output.

F.4 RETRIEVER f_γ

You are a text retriever.
You are given a question and a set of memory content indices. Each index is a brief summary of the key information in the corresponding memory content.
Your task is to, based on the given question, identify the ids of the memory indices that are most likely to provide context for answering the question.
Example:
Question: Where is Alice's home?
Indices:
id:0, dialogue time:13 October, 2022, Alice's two children
id:1, dialogue time:13 October, 2023, Alice's husband
id:2, dialogue time:23 October, 2022, Jack's job
id:3, dialogue time:13 October, 2022, Charity organization
id:4, dialogue time:31 October, 2022, Alice moved from her hometown
id:5, dialogue time:31 October, 2022, Alice's life in her hometown
Result:
{ "keywords_list": [4,5] }

F.5 GENERATOR G_γ

You are a memory-based question-answering assistant. You will receive a question along with memories for answering it.
Your main responsibilities are as follows:
Please note that answers to questions are not always fixed. You should list all possible answers and strive to ensure the completeness of the information. For example, regarding time-related questions, to avoid ambiguity, explicitly specify the reference point for any relative time expressions. For instance, if the answer is "last year," the correct format should be: "The current year is 2022, and the answer is last year." At the same time, not all questions have relevant memories, such as open-ended questions. Even if the memory information is incomplete, you can boldly infer the most likely answer based on the existing memories, even if the answer may be incomplete or not entirely rigorous. Avoid refusing to answer.
The required output format is: { "answer": "..." }
Note: Only return your output in JSON format, and do not provide any other form of output.

F.6 LLM AS JUDGE

You are an answer scoring expert. You will receive the following information: (1) a question, (2) a standard (reference answer), (3) an answer generated by a memory-based LLM. Your task is to label the generated answer as CORRECT or WRONG. Important notes: Be as lenient as possible when scoring: the standard answer is typically a concise short sentence, while the generated answer may be longer and more detailed. As long as the generated answer aligns with the standard answer, it should be marked as CORRECT. Do not be overly nitpicky. For time-related questions, the standard answer is a specific date, month, or year. The generated answer may be longer or use relative time expressions (e.g., "last Tuesday" or "next month"). Please be lenient in scoring—if the generated answer refers to the same time period as the standard answer, mark it as CORRECT. Finally, provide CORRECT or WRONG. Do not include both CORRECT and WRONG in your response, as this will cause errors in the evaluation script. Simply return the label in JSON format with the key "label". Now it's time for the formal question: Question: {question} Gold answer: {gold_answer} Generated answer: {generated_answer}