Multi-Head Recurrent Memory Agents

Jiatong Li **Samuel Yeh** **Sharon Li**
Department of Computer Science, University of Wisconsin-Madison
{jli2947, samuelyeh, sharonli}@cs.wisc.edu

Abstract

Recurrent memory agents extend LLMs to arbitrarily long contexts by iteratively consolidating input into a fixed-size memory window. Despite their scalability, these agents exhibit a well-documented reliability problem: end-to-end performance degrades systematically as context length grows. We diagnose this failure by decomposing performance into two factors—memory *capture* and memory *retention*—and quantitatively confirm that retention is the dominant bottleneck. Retention collapses because existing designs maintain memory as a monolithic text block, forcing every update to risk overwriting previously retained content. Motivated by this diagnosis, we propose **Multi-Head Recurrent Memory (MHM)**, a general, training-free framework that partitions memory into independent heads governed by a stage-wise select-then-update strategy. At each step, exactly one head is selected for update while the remaining heads are structurally shielded from overwriting, shifting the burden of retention from model behavior to architectural design. As a lightweight instantiation, we introduce Least-Recently-Updated MHM (MHM-LRU), which guarantees uniform head utilization with zero additional token overhead. Extensive experiments on long-context benchmarks show that MHM-LRU substantially improves both retention and end-to-end accuracy across the 100K–1M token range, where baselines degrade sharply. On RULER-HQA at 896K tokens, MHM-LRU improves the memory retention rate from less than 30% to **73.96%**. These gains generalize across model families, scales, and task types, positioning architectural optimization as a practical and cost-efficient path toward reliable long-context recurrent memory. Code: https://github.com/deeplearning-wisc/multi-head-memory

1 Introduction

The ability to reason over long contexts is a fundamental requirement for LLM agents operating in the real world [1–5]. Tasks such as document analysis [6–9], multi-step research [10–12], and extended dialogue [1, 13] demand that agents track and integrate information across inputs that far exceed the context window. Recurrent memory [14–16] addresses this challenge by processing input sequentially in chunks and iteratively compressing a long context into a fixed-size memory window. At each step, the agent reads a new context chunk and updates its memory to integrate new information with what has been retained so far. This paradigm is appealing because it places no hard limit on context length and requires no modification to the underlying LLM. In practice, however, recurrent memory agents exhibit a well-documented reliability problem: performance degrades systematically as context length grows [17, 18].

To motivate our work, we first diagnose how existing recurrent memory fails under long contexts by decomposing end-to-end performance into two factors: whether the memory ever *captures* the relevant information during memory update, and whether it *retains* that information through all subsequent memory updates. Measuring both memory capture rate and memory retention rate on a representative method MemAgent [17], we find that capture remains stable across all context

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

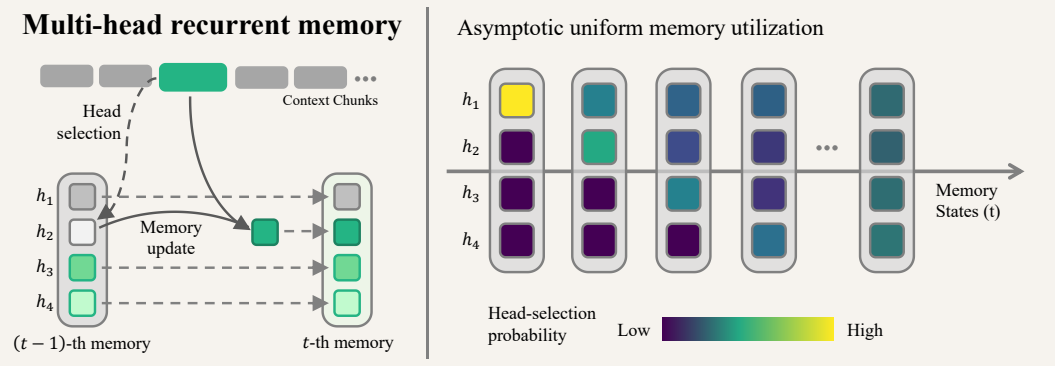


Figure 1: Left: An overview of the multi-head recurrent memory (MHM). Right: An overview of the memory head utilization pattern across time. In every time step, only one head is written. As time goes by, MHM achieves asymptotic uniform memory utilization.

lengths, while memory retention decreases sharply, falling below 30% at 896K tokens. This diagnosis pinpoints memory retention failure as the dominant bottleneck. The root cause is architectural: existing recurrent memory designs maintain memory as a monolithic text block, so every update risks erasing key information. Information written at step t must survive $(T - t)$ subsequent overwrites to reach the final state, an increasingly unlikely outcome as T grows.

Motivated by the diagnosis, we propose **Multi-Head Recurrent Memory (MHM)**, a general, training-free memory framework designed to target memory retention failure. Rather than maintaining a single monolithic memory block, MHM partitions memory into multiple independent heads and governs updates through a stage-wise select-then-update strategy. At each step, one of the heads is selected for update while the remaining heads are held fixed. This shifts the burden of retention from model behavior to the architectural design: rather than relying on the model to avoid overwriting important content, MHM structurally shields the information retained in unselected heads from erasure at each step. The selection criterion is left general, accommodating any rule-based or model-based strategy, and existing single-headed methods are subsumed as the special case.

As a lightweight instantiation, we showcase MHM-LRU, which selects the least-recently-updated head at each step. To diversify memory head use to persist memory retention, this rule provably guarantees uniform head utilization across heads and incurs zero additional token overhead relative to a single-headed baseline. Extensive experiments on realistic long-context benchmarks demonstrate that MHM-LRU substantially improves memory retention, outperforming baselines across all evaluated context lengths. Notably, on RULER-HQA at 896K tokens, MHM-LRU improves memory retention rate from less than 30% to **73.96**%. The retention improvements directly translate into end-to-end performance gains, where MHM-LRU's accuracy remains strong and stable across the 100K-1M context range while baselines degrade sharply. These benefits generalize across model families, scales, and task types, positioning architectural optimization as a practical and cost-efficient path toward reliable long-context recurrent memory. Overall, the contributions of this paper are summarized as follows:

1. We identify memory retention failure as the dominant bottleneck of recurrent memory agents under extremely long contexts, supported by a quantitative capture-retention decomposition analysis.
2. We propose Multi-Head Recurrent Memory (MHM), a general and training-free memory architecture-level framework that structurally limits overwrite pressure through independent memory heads and stage-wise selective updates.
3. We extensively validate MHM on realistic long-context benchmarks, demonstrating substantial retention and accuracy improvements across context lengths, model families, and task types.

2 Preliminaries and Related Work

Long-Context LLMs. Scaling LLMs to long contexts has been pursued along two primary directions at the model level: extending the context window via positional encoding modifications [19–21]

and reducing attention complexity via sparse or linear attention mechanisms [22–24]. These advances have enabled million-token-level context windows in both open-source models such as Qwen2.5-1M [25], and proprietary systems such as Claude-Sonnet-4.6 [26] and Gemini-3.1-Pro [27]. Despite their effectiveness, native long-context LLMs remain bounded by fixed context limits or quadratic memory costs, motivating recurrent memory as a complementary paradigm that sidesteps both constraints.

Recurrent Memory Agents. Early studies in agent memory [28–30] explore using an external memory store and retrieval to address long-term workflows. In contrast, recurrent memory appears to be targeted for single long-context processing of LLMs by recurrently maintaining a memory state. The idea of recurrent memory was foreshadowed in earlier architecture-level approaches [31–34]. However, constrained by the representational capacity of the underlying architecture, the generality of these methods are limited in a broader range of applications. It was not until the emergence of token-level recurrent memory [17, 14] that this paradigm was concretely established.

Formally, let (C, q) denote a context-query pair where an LLM agent must produce a correct response y. Since C may far exceed the model's context window, the agent partitions C into an ordered chunk sequence $C = (c_1, c_2, \ldots, c_T)$ and maintains a bounded memory state m_t through sequential updates. The memory agent f_θ then processes these chunks sequentially, updating a bounded memory state m_t at each step:

$$m_t \leftarrow f_\theta(m_{t-1},\, c_t,\, q), \quad t = 1, 2, \ldots, T. \tag{1}$$

After all chunks have been processed, the LLM agent generates its final response conditioned solely on the accumulated memory:

$$y \leftarrow \text{LLM}(m_T,\, q). \tag{2}$$

Because m_T is the sole information carrier from the first T chunks to the final response, the fidelity of the update in Eq. (1) under growing T is the central reliability challenge of this paradigm. This paradigm has been concretely established by MemAgent [17] for long-context query answering. Subsequent work has extended it in several directions: GRU-Mem [35] introduces text-controlled gating for selective updates, while ReMemR1 [36] augments recurrent memory with a revisitable retrieval component, yet the core recurrent working memory remains a monolithic block.

Single-Headed Memory as a Structural Bottleneck. Across existing recurrent memory designs [17, 36, 35], the core recurrent memory state m_t is maintained as a single monolithic text block—a design we term *single-head* memory. At every update step, the memory agent must overwrite this block with content derived from c_t, creating an inherent tension between retaining previously captured information and incorporating new context. This tension compounds with context length: information captured at step t must survive $(T - t)$ subsequent overwrites intact to reach m_T. Under sufficiently long contexts, compression into a single representational slot makes information loss structurally inevitable. In the next section, we rigorously characterize and measure this failure, identifying it as the binding constraint on long-context performance.

3 Characterizing Recurrent Memory Failure

To motivate our memory architectural design, we first analyze *how* recurrent memory fails under long contexts. Incentivized by the working memory [37] and memory forgetting detection [38, 39] in neuroscience, we decompose end-to-end performance into two memory factors: *capture* and *retention*. For a correct response, the memory must (1) *capture* the relevant information at some point during iterative updates, and (2) *retain* it through all subsequent updates until the final state m_T. Failure at either stage produces a wrong answer, but the two modes need not be equally prevalent in practice.

Definition 1 (**Memory Capture Rate (MCR)**). *Let $\mathcal{D} = \{(C_i, q_i, y_i^*)\}_{i=1}^N$ be an evaluation dataset and $\mathcal{M} = \{m_i\}_{i=1}^N$ the corresponding memory trajectories. The $m_i = \{m_{i,1}, \ldots, m_{i,T}\}$ denotes the memory state trajectory, where T is the total number of trajectory steps for the respective instance. We assess answer membership $\mathbb{1}(y_i^* \in m_{i,t})$ of the ground-truth answer against the memory content at step t. MCR is defined as the fraction of queries for which the relevant information was present in*

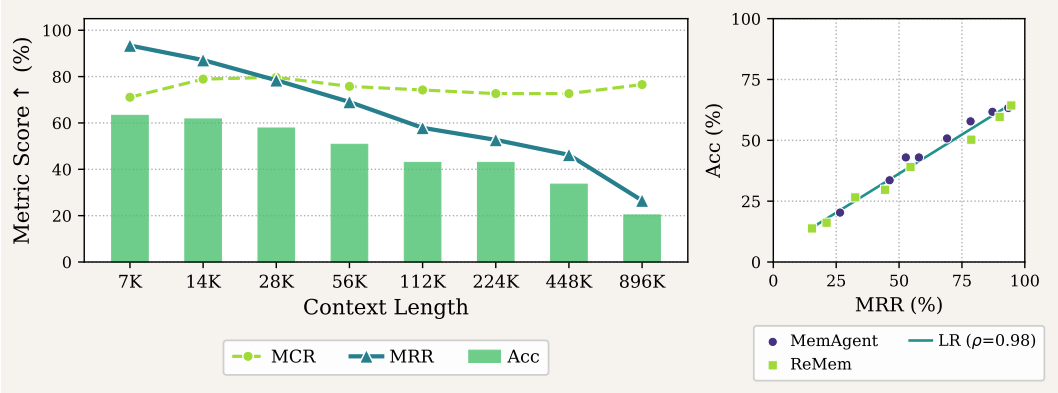


Figure 2: **(a)** MCR and MRR of MemAgent [17] (Qwen2.5-14B-Instruct) on RULER-HQA across context lengths from 7K to 896K tokens. While MCR remains stable throughout, MRR degrades sharply with context length, falling below 30% at 896K tokens. **(b)** MRR and Acc correlation analysis of MemAgent and ReMem [36] on RULER-HQA across context lengths from 7K to 896K tokens. The 'LR' denotes linear regression. End-to-end accuracy correlates strongly with MRR ($\rho = 0.98$), confirming that retention failure is the dominant factor.

memory at least once during processing:

$$\mathrm{MCR}(\mathcal{M}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i^* \in m_{i,t},\ \exists\, t \in [T])\,. \tag{3}$$

Definition 2 (**Memory Retention Rate (MRR)**). *Among queries where memory capture occurred, the fraction for which the captured information survived through to the final memory state m_T:*

$$\mathrm{MRR}(\mathcal{M}, \mathcal{D}) = \frac{\sum_{i=1}^{N} \mathbb{1}(y_i^* \in m_{i,t}, \exists\, t \in [T-1]) \cdot \mathbb{1}(y_i^* \in m_{i,T})}{\sum_{i=1}^{N} \mathbb{1}(y_i^* \in m_{i,t},\ \exists\, t \in [T])}. \tag{4}$$

A high MRR indicates that the memory agent is capable of retaining previously captured information when processing irrelevant context.

Retention Failure Dominates. We apply this analysis to MemAgent, a representative recurrent memory baseline built on Qwen2.5-14B-Instruct [40], on the RULER-HQA [41, 42] benchmark. As shown in Figure 2 (left), MCR remains stable across all evaluated context lengths, indicating that the agent consistently captures relevant information regardless of input length. MRR, by contrast, degrades sharply, falling below 30% at 896K tokens. Since MCR is roughly stable, the observed accuracy degradation is almost entirely attributable to retention failure. Appending results from ReMem [36], Figure 2 (right) further confirms that end-to-end accuracy correlates strongly with MRR, with correlation coefficient $\rho = 0.98$. This isolates memory retention as the main structural bottleneck, and directly motivates the design question we address in Section 4: how to make the memory structure itself resistant to overwrite pressure.

4 Multi-Head Recurrent Memory

Motivated by our analysis, we propose **Multi-Head Recurrent Memory** (**MHM**), a general framework designed to alleviate retention failure. The core insight is that retention failure is not a failure of the model but the memory structure: when a single block holds all context, every update is a potential erasure. MHM addresses this at the source by splitting memory into H mutually independent memory heads and governing updates through a selective overwrite strategy. More specifically, MHM represents the memory state as a set of H independent text blocks, $m_t = \{m_t^1, \ldots, m_t^H\}$, where each head m_t^h is delimited by `<memory_{h}>` ... `</memory_{h}>` tags. Each update affects only one memory head, while the remaining $(H-1)$ heads are structurally shielded from erasure by construction. Our design addresses two challenges: which head to select for update, and how to perform the update given the selected head. We describe the two stages below.

Selection Stage. The selection stage addresses the memory retention failure through *diversified* head selection—by spreading updates across different heads over time, the agent avoids repeatedly overwriting the same head and thereby reduces the risk of erasing pivotal information. Formally, given the current memory state $m_{t-1} = \{m_{t-1}^1, \ldots, m_{t-1}^H\}$, the selection stage identifies the head $h^\star \in [H]$ most worthy of being updated:

$$h^\star = \arg\max_{h \in [H]} \; r_{\text{select}}(h \mid m_{t-1}, q, c_t). \tag{5}$$

where $r_{\text{select}}(h \mid m_{t-1}, q, c_t)$ is a criterion function that assigns a priority score to each head candidate. The criterion r_{select} can encode a wide range of selection strategies depending on the available information and computational budget. A rule-based criterion may select based on head age or a fixed schedule. A model-based criterion may assess the relevance of each head given the current chunk and query. Regardless of the specific instantiation, a well-designed criterion should satisfy a key property: *diversity*, ensuring that updates are spread across heads over time so that no single head bears disproportionate overwrite pressure. We will provide an ablation on different selection strategies in Section 5.3.

Update Stage. The update stage overwrites the selected head $h^\star$ with content derived from the current context chunk. Formally, given the current memory state m_{t-1}, the selected head index $h^\star$, the query q, and the current chunk c_t, the memory agent generates new head content and updates the memory state as follows:

$$m_t^{h^\star} = f_\theta(m_{t-1},\, h^\star,\, q,\, c_t), \qquad m_t^h = m_{t-1}^h \;\; \forall\, h \neq h^\star. \tag{6}$$

By receiving the full memory state m_{t-1} as context, the update allows the memory agent to reason jointly about what has already been retained across all heads and what new information from c_t is worth writing. This encourages both capturing newly relevant content and avoiding redundancy with information already preserved in unselected heads. This stage-wise design subsumes existing single-headed methods as the special case $H = 1$, making MHM a flexible and general framework.

A Lightweight Instantiation. As a concrete instantiation of MHM, inspired by the least-recently-used cache replacement algorithm [43–45], we adopt a least-recently-updated (LRU) selection rule: at each step, the head that has gone the longest without an update is selected. This instantiation has two advantages. First, the selection strategy ensures diversified head selection. When all heads are initialized identically, this provides a structural guarantee that no head is updated more than once before all others are updated at least once. This ensures uniform utilization across heads and maximizes memory redundancy. Second, this method is computationally lightweight since head selection requires only a simple age comparison rather than an additional LLM inference pass, incurring *no extra token overhead* relative to a single-headed baseline. We refer to this instantiation as **MHM-LRU**. As our experiments demonstrate, the structural guarantee of uniform head utilization can substantially improve retention and end-to-end performance in long context scenarios, suggesting that the benefits of MHM do not depend on sophisticated selection strategies. Full prompt templates and implementation details are provided in Appendix A.2.

5 Experiments

5.1 Experimental Setup

Baselines. We compare MHM-LRU against two latest recurrent memory baselines: (1) MemAgent [17], which maintains a single memory head and applies a full overwrite at every update step, and (2) ReMem [36], which maintains two heterogeneous heads (an update memory and a retrieval memory) where the retrieval memory is refreshed via a rule-based callback query. GRU-Mem [35] is excluded as its implementation is not publicly available. All methods use Qwen2.5-14B-Instruct as the backbone of the recurrent memory structure. We additionally evaluate on Qwen2.5-32B-Instruct and gpt-oss-120b in the ablation study. To ensure fair comparison, ***the total memory capacity is kept the same across all methods***: 4,096 tokens for MemAgent (one head), 2,048 per head for ReMem (two heads), and 1,024 per head for MHM-LRU (4 heads). Full decoding and evaluation settings are provided in Appendix B.

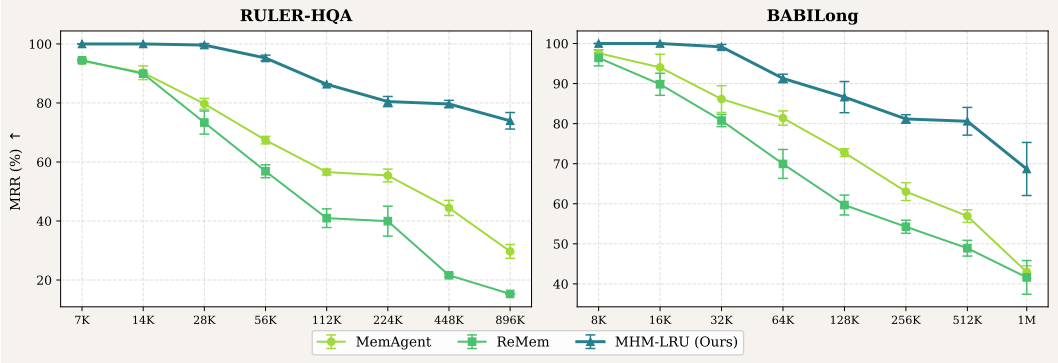


Figure 3: MRR results across different context lengths. The x-axis represents context length. Each data point is an average across 3 random runs with standard deviation as the error bar.

Table 1: End-to-end answer accuracy↑ (%) results on RULER-HQA.

Method	Context Length							
	7K	**14K**	**28K**	**56K**	**112K**	**224K**	**448K**	**896K**
LLM Baseline	**68.75**	**65.62**	56.25	**56.25**	46.88	19.53	10.94	3.91
MemAgent	62.24±0.97	60.94±1.10	56.25±2.21	50.52±0.37	42.19±0.64	42.97±0.64	33.33±0.98	21.62±0.98
ReMem	64.32±3.51	59.64±0.98	50.26±3.68	39.06±2.78	29.68±2.78	26.56±1.69	16.14±1.33	13.80±0.97
MHM-LRU	66.86±1.69	62.76±1.47	**59.64±1.33**	55.21±1.33	**50.26±1.47**	**50.26±2.57**	**48.70±2.88**	**49.74±1.47**

Datasets. We evaluate on two benchmarks: (1) `RULER-HQA` [17], a multi-hop QA dataset spanning context lengths from 7K to 892K tokens, following the evaluation protocol of [17], and (2) `BABILong` [46], a long-context reasoning benchmark covering ten structurally distinct task types (Q1–Q10) across context lengths from 1K to 10M tokens. For BABILong, we evaluate on context lengths from 8K to 1M tokens. For each length, we sample 128 entries via systematic sampling to ensure uniform coverage of all task types. All recurrent memory methods are run three times independently; we report the mean and standard deviation throughout.

5.2 Main Results

MHM Substantially Improves Memory Retention Figure 3 shows MRR across context lengths for all methods. We observe that MHM-LRU consistently and substantially outperforms both baselines, with the gap widening at longer contexts. For example, on RULER-HQA at 896K tokens, MHM-LRU achieves an MRR of **73.96**% compared to less than 30% for MemAgent and ReMem. On BABILong at 1M tokens, MHM-LRU reaches 68.68% versus 42.96% for MemAgent. These results demonstrate the effectiveness of MHM-LRU in alleviating the memory retention issue of recurrent memories as context length increases. Moreover, at shorter context lengths (under 50K tokens), MHM-LRU achieves near-perfect retention (MRR $>$ 99%) on both datasets, confirming that the multi-head design incurs no penalty in the common short-to-mid-length regime. Overall, the experiments demonstrate the effectiveness of MHM in addressing the retention of current memory.

Table 2: End-to-end answer accuracy↑ (%) results on BABILong.

Method	Context Length							
	8K	**16K**	**32K**	**64K**	**128K**	**256K**	**512K**	**1M**
LLM Baseline	**64.06**	57.81	53.12	**53.12**	**50.0**	35.16	31.25	27.34
MemAgent	58.85±2.66	55.99±2.58	53.39±2.42	47.14±0.98	44.66±1.64	38.02±2.88	32.55±1.61	25.26±0.97
ReMem	61.46±2.41	55.21±2.24	45.57±1.61	42.19±3.99	34.64±2.42	32.29±1.61	29.50±0.26	24.48±0.97
MHM-LRU	63.54±1.60	**60.94±2.92**	**55.47±1.27**	51.82±0.97	48.70±3.27	**47.39±1.84**	**46.35±2.66**	**41.41±1.91**

Retention Gains Translate Directly to End-to-End Performance. Table 1 and Table 2 report end-to-end accuracy across all context lengths. We observe that MHM-LRU consistently outperforms both recurrent memory baselines and the native long-context LLM across nearly all settings. The advantage is most pronounced at extreme context lengths: on RULER-HQA at 896K tokens, MHM-LRU achieves 49.74% accuracy versus 21.62% for MemAgent and 3.91% for the LLM baseline; on BABILong at 1M tokens, 41.41% versus 25.26% for MemAgent. Crucially, MHM-LRU's accuracy remains stable across the 100K–1M range (48.70% – 50.26% on RULER-HQA; 41.41% – 48.70% on BABILong), whereas MemAgent and ReMem degrade sharply. Taken together with the MRR results in Figure 3, these findings confirm that the retention improvements from MHM directly translate into downstream performance gains.

5.3 Ablation Studies on Key Design

We ablate two design dimensions of MHM: the number of heads and the memory selection strategy. All ablations are conducted on RULER-HQA using Qwen2.5-14B-Instruct, with per-head capacity fixed at 1,024 tokens.

More Memory Heads Improve Retention. We evaluate the impact of the number of heads H on the memory retention rate (MRR). We vary H from 1 to 16, under each context length. As shown in Figure 4, MRR shows approximately logistic-like improvement as the number of heads grows. For example, under context length 896K, our approach improves MRR from 26.53% with one head (equivalent to MemAgent) to 69.89% with two heads, 74.67% with four heads, and 87.12% with eight heads, before plateauing at sixteen heads. This trend holds consistently across all evaluated context lengths. This ablation study reveals that increasing the number of heads can significantly improve memory retention, confirming the structural advantage of our design.

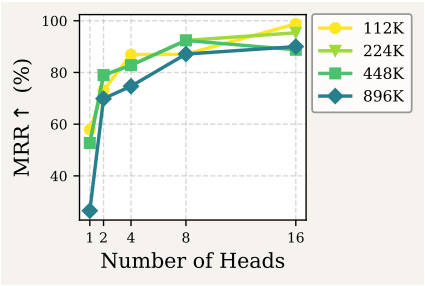


Figure 4: Effect of number of memory heads across context lengths.

Stage-wise Selection Reduces Retention Failure. A critical design in MHM is the stage-wise head selection and memory update. We thus compare MHM-LRU against a variant MHM-Concur to isolate the effect of the stage-wise design. In MHM-Concur, head selection and memory update are merged into a single step: the model is asked to simultaneously decide which head to overwrite and produce the updated content. As shown in Table 3, while MHM-Concur achieves a notably higher MCR than the stage-wise approach, it suffers from substantially lower MRR (54.12% vs. 73.96% at context length 896K), resulting in lower end-to-end accuracy across all context lengths. The concurrent design introduces instruction complexity that degrades the model's ability to retain captured content: when the model must simultaneously reason about head routing and content generation, it is more likely to inadvertently erase previously retained information. In contrast, the stage-wise design eliminates this conflict by dedicating separate prompts to each operation, allowing the update stage to focus entirely on capture and retention without the competing objective of head routing.

Table 3: Ablation study of the multi-headed memory selection strategy.

Metric	Context Length			
	112K	**224K**	**448K**	**896K**
MHM-Concur (w/o stage-wise)				
Acc↑ (%)	46.88	42.97	35.94	33.59
MCR↑ (%)	71.09	74.22	74.22	66.41
MRR↑ (%)	72.53	73.68	56.81	54.12
MHM-Relevance				
Acc↑ (%)	46.09	50.78	39.84	45.31
MCR↑ (%)	69.53	70.31	65.62	67.97
MRR↑ (%)	82.02	78.89	70.24	74.71
MHM-LRU				
Acc↑ (%)	50.26	50.26	48.70	49.74
MCR↑ (%)	67.97	71.88	71.09	69.16
MRR↑ (%)	86.34	80.48	79.66	73.96

Rule-based vs. Relevance-based Selection. We compare MHM-LRU (rule-based) against another variant, MHM-Relevance, to isolate the effect of the selection criterion, holding the stage-wise design fixed. In MHM-Relevance, the model is explicitly prompted to select the head most worthy of being updated, while diversifying the head usage (details in Appendix A.5). The intuition is that the model can make relevance-aware routing decisions. Table 3 shows that, despite this appeal, MHM-

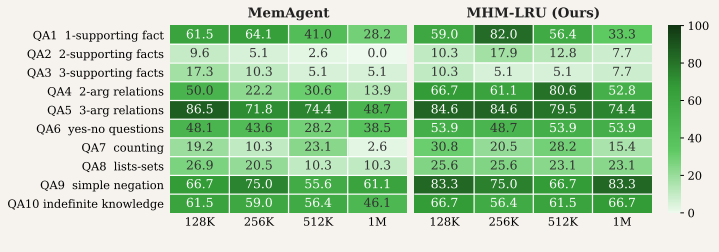


Figure 5: Accuracy by task type (QA1 – QA10) and context length (128K – 1M) on BABILong.

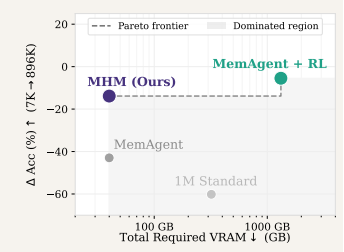


Figure 6: VRAM cost and accuracy change on RULER-HQA.

Relevance usually incurs sub-optimal MCR and MRR, leading to lower end-to-end accuracy (45.31% vs. 49.74% at context length 896K). The model-driven routing has two compounding issues. First, model-based relevance assessment at each step is inherently noisy: the model cannot reliably predict which information will be needed at query time, leading to routing decisions that concentrate updates on heads perceived as salient and starve others of updates. Second, this concentration undermines coverage diversity, reducing the breadth of information captured across heads. MHM-LRU sidesteps the issues by guaranteeing uniform head utilization without any model involvement, requires no additional inference for selection, and achieves overall better performance.

5.4 Generality Analysis

Generality Across Model Family and Scale. We evaluate MHM-LRU with two additional base models: Qwen2.5-32B-Instruct and gpt-oss-120b. We focus on the 112K–896K settings, where retention failure is most severe. Across both models, MHM-LRU maintains significantly higher and more stable MRR than MemAgent, as shown in Table 4. On gpt-oss-120b, MemAgent's MRR falls below 40%, while MHM-LRU maintains MRR to 68.85%. On Qwen2.5-32B-Instruct, MHM-LRU's MRR consistently achieves 90% on 112K–448K and maintains to 81.91% on 896K. This consistent pattern across model families and scales confirms that the retention benefit of MHM does not depend on any model-specific property, but follows from the memory architectural advantage.

Table 4: MRR (%) of different model families and scales. Best metric value given the model and context length is **bolded**.

Model	Memory	Context Length			
		112K	224K	448K	896K
gpt-oss-120b	MemAgent	75.00	55.84	61.64	35.71
	MHM-LRU	**84.48**	**75.44**	**69.35**	**68.85**
Qwen2.5-32B	MemAgent	84.04	87.63	88.54	78.02
	MHM-LRU	**93.02**	**95.65**	**90.00**	**81.91**

Generality Across Task Types. To evaluate the generality of MHM-LRU across diverse task types, we conduct experiments on BABILong, which encompasses ten structurally distinct reasoning tasks (QA1–QA10). As shown in Figure 5, MHM-LRU outperforms MemAgent across all ten task types at long contexts (1M), with particularly pronounced gains on tasks requiring relational reasoning and yes-no judgment (QA4–QA6), where MemAgent's accuracy degrades sharply with context length. This pattern aligns with our retention analysis: tasks involving multi-step relational chains are most susceptible to memory retention failure, making them the primary beneficiaries of MHM's structural redundancy. These results confirm that MHM-LRU's advantage is not task-specific but reflects a systematic improvement that generalizes across task types.

Table 5: Time and output token costs of recurrent memory agents on RULER-HQA. For each method, the test is run on an NVIDIA A100 80GB, with four parallel threads.

Metric | Method		7K	14K	28K	56K	112K	224K	448K	896K
Time / Sample (sec)	MemAgent	2.8	5.1	9.2	17.3	31.5	59.1	114.3	224.9
	MHM-LRU	1.9	3.3	6.5	13.0	25.9	53.4	117.1	239.1
Tokens / Sample	MemAgent	330.6	535.8	916.2	1705.6	2834.4	4943.4	9340.6	18013.9
	MHM-LRU	142.8	214.4	419.5	898.9	1875.7	3814.0	9038.6	18085.5

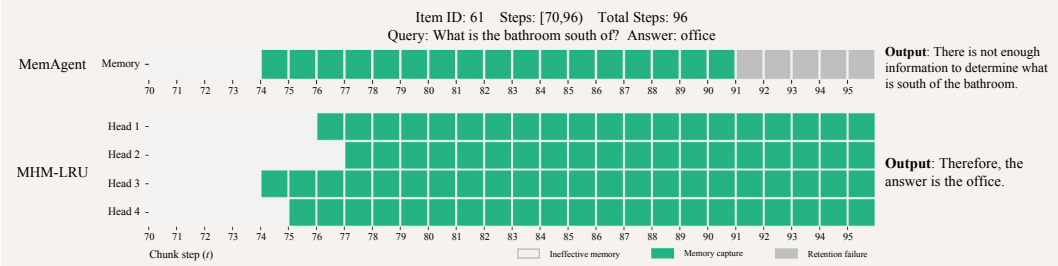


Figure 7: A case study of memory trajectory on BABILong-512K. The task type is QA4: 2-arg relations.

Computational Efficiency. Figure 6 plots accuracy degradation (7K to 896K) against total VRAM cost for all methods. We additionally include MemAgent+RL (RL-MemAgent [17]), a reinforcement learning post-trained variant of MemAgent that requires significantly more VRAM due to the overhead of RL training. MHM-LRU lies on the Pareto frontier between these two extremes: it recovers a substantial fraction of the accuracy gain achieved by MemAgent+RL at no additional memory overhead beyond vanilla MemAgent. This is notable because MHM-LRU requires neither model weight access nor task-specific training, and the gain comes entirely from the architectural design with multiple heads. This positions architectural optimization as a practical and resource-efficient path toward reliable long-context recurrent memory, particularly in settings where model retraining or large VRAM budgets are unavailable.

We further evaluate the time and output token costs of different recurrent memory methods under long contexts, as shown in Table 5. MHM-LRU is consistently faster than MemAgent at short-to-mid context lengths (e.g., 25.9s vs. 31.5s at 112K), with negligible overhead at extreme lengths. Similarly, MHM-LRU consumes substantially fewer output tokens at shorter contexts (142.8 vs. 330.6 at 7K), converging to parity with MemAgent only at 896K. Both effects reflect a spontaneous amortization property of MHM: by writing to a single head per step, the LLM naturally generates fewer tokens per update, amortizing the cost of memory maintainance. These results confirm that MHM-LRU achieves its retention and accuracy gains at no meaningful increase in computational cost.

5.5 Memory Dynamics Case Study

To concretely illustrate how MHM-LRU alleviates retention failure, Figure 7 presents a representative memory trajectory on a BABILong-512K sample from the 2-arg relations task. The query asks "What is the bathroom south of?" (ground truth: "office"), with relevant chunks falling within steps [70, 96). MemAgent captures the answer from step 74 onward but overwrites it during the final updates (steps 91–95), yielding the incorrect response "There is not enough information to determine what is south of the bathroom.", which is a representative instance of retention failure. MHM-LRU, in contrast, captures the answer at step 74 and then spontaneously propagates it to the remaining heads in subsequent updates, maintaining multiple redundant copies that persist through step 95 and yielding the correct answer "Therefore, the answer is the office." This emergent backup behavior concretely demonstrates how partitioning memory into independent heads converts the stochastic overwrite risk of a monolithic block into an architectural retention guarantee.

6 Conclusion and Limitation

We presented Multi-Head Recurrent Memory (MHM), a new framework for improving the reliability of recurrent memory agents under long contexts. Through a diagnostic analysis of memory dynamics, we identified retention failure as the dominant bottleneck of long-context performance. Motivated by this diagnosis, MHM is designed to address the memory retention failure through multiple memory heads and the stage-wise memory update strategy. MHM structurally limits overwrite pressure on retained content without requiring any model retraining. MHM-LRU, a lightweight instantiation, substantially improves memory retention and end-to-end accuracy on challenging long-context query benchmarks. We hope these findings encourage future work on architectural approaches to long-context reliability, and on combining MHM with training-based methods for further gains.

Limitation. In our experiments, the number of heads H is fixed. While our ablation shows that MRR improves monotonically with H, the optimal number of heads is likely dependent on context length and task structure. An adaptive head allocation strategy that adjusts H based on input characteristics is a natural direction for future work.

Acknowledgment

We sincerely thank Changdae Oh, Linyue Cai and Hyeong Kyu Choi for proofreading the manuscript, and Jimmy Di for providing additional computational resources. This work is supported in part by the AFOSR Young Investigator Program under award number FA9550-23-1-0184, Office of Naval Research under award N000142612508, National Science Foundation under awards IIS-2237037 and IIS-2331669, Schmidt Sciences Foundation, Open Philanthropy (now Coefficient Giving), Alfred P. Sloan Fellowship, and gifts from Google and Amazon.

References

[1] Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arık. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024.

[2] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, 2024.

[3] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can context length of open-source llms truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[4] Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. Loogle: Can long-context language models understand long contexts? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16304–16333, 2024.

[5] Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *Advances in neural information processing systems*, 37:119638–119661, 2024.

[6] Minzheng Wang, Longze Chen, Fu Cheng, Shengyi Liao, Xinghua Zhang, Bingli Wu, Haiyang Yu, Nan Xu, Lei Zhang, Run Luo, et al. Leave no document behind: Benchmarking long-context llms with extended multi-doc qa. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5627–5646, 2024.

[7] Yubo Ma, Yuhang Zang, Liangyu Chen, Meiqi Chen, Yizhu Jiao, Xinze Li, Xinyuan Lu, Ziyu Liu, Yan Ma, Xiaoyi Dong, et al. Mmlongbench-doc: Benchmarking long-context document understanding with visualizations. *Advances in Neural Information Processing Systems*, 37:95963–96010, 2024.

[8] Xinze Li, Yushi Bai, Bowen Jin, Fengbin Zhu, Liangming Pan, and Yixin Cao. Long context vs. rag: Strategies for processing long documents in llms. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4110–4113, 2025.

[9] Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang 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. *Arxiv:2509.13313*, 2025.

[10] Yuxuan Huang, Yihang Chen, Haozheng Zhang, Kang Li, Huichi Zhou, Meng Fang, Linyi Yang, Xiaoguang Li, Lifeng Shang, Songcen Xu, et al. Deep research agents: A systematic examination and roadmap. *Arxiv:2506.18096*, 2025.

[11] Weizhi Zhang, Yangning Li, Yuanchen Bei, Junyu Luo, Guancheng Wan, Liangwei Yang, Chenxuan Xie, Yuyao Yang, Wei-Chieh Huang, Chunyu Miao, et al. From web search towards agentic deep research: Incentivizing search with reasoning agents. *Arxiv:2506.18959*, 2025.

[12] Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, Kuan Li, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Iterresearch: Rethinking long-horizon agents via markovian state reconstruction. In *The Fourteenth International Conference on Learning Representations*, 2026.

[13] 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)*, pages 8416–8439, 2025.

[14] Wangchunshu Zhou, Yuchen Eleanor Jiang, Peng Cui, Tiannan Wang, Zhenxin Xiao, Yifan Hou, Ryan Cotterell, and Mrinmaya Sachan. Recurrentgpt: Interactive generation of (arbitrarily) long text. *Arxiv:2305.13304*, 2023.

[15] Qianhao Yuan, Jie Lou, Zichao Li, Jiawei Chen, Yaojie Lu, Hongyu Lin, Le Sun, Debing Zhang, and Xianpei Han. Memsearcher: Training llms to reason, search and manage memory via end-to-end reinforcement learning. *Arxiv:2511.02805*, 2025.

[16] 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 *The Fourteenth International Conference on Learning Representations*, 2026.

[17] 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 *The Fourteenth International Conference on Learning Representations*, 2026.

[18] Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Babu Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A. Huerta, and Hao Peng. Context length alone hurts LLM performance despite perfect retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23281–23298, 2025.

[19] Amirhossein Kazemnejad, Inkit Padhi, Karthikeyan Natesan, Payel Das, and Siva Reddy. The impact of positional encoding on length generalization in transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[20] Zhenyu He, Guhao Feng, Shengjie Luo, Kai Yang, Liwei Wang, Jingjing Xu, Zhi Zhang, Hongxia Yang, and Di He. Two stones hit one bird: Bilevel positional encoding for better length extrapolation. In *Forty-first International Conference on Machine Learning*, 2024.

[21] Tong Wu, Yanpeng Zhao, and Zilong Zheng. An efficient recipe for long context extension via middle-focused positional encoding. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[22] Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *The Thirteenth International Conference on Learning Representations*, 2025.

[23] Shang Yang, Junxian Guo, Haotian Tang, Qinghao Hu, Guangxuan Xiao, Jiaming Tang, Yujun Lin, Zhijian Liu, Yao Lu, and Song Han. LServe: Efficient long-sequence LLM serving with unified sparse attention. In *Eighth Conference on Machine Learning and Systems*, 2025.

[24] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

[25] An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1m technical report. *Arxiv:2501.15383*, 2025.

[26] Anthropic. Introducing claude sonnet 4.6, 2026.

[27] Google. Gemini 3.1 pro: A smarter model for your most complex tasks, 2026.

[28] 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.

[29] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

[30] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

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

[32] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *Arxiv:2312.00752*, 2023.

[33] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, et al. Rwkv: Reinventing rnns for the transformer era. In *Findings of the association for computational linguistics: EMNLP 2023*, pages 14048–14077, 2023.

[34] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *Arxiv:2501.00663*, 2024.

[35] Leheng Sheng, Yongtao Zhang, Wenchang Ma, Yaorui Shi, Ting Huang, Xiang Wang, An Zhang, Ke Shen, and Tat-Seng Chua. When to memorize and when to stop: Gated recurrent memory for long-context reasoning. *Arxiv:2602.10560*, 2026.

[36] Yaorui Shi, Yuxin Chen, Siyuan Wang, Sihang Li, Hengxing Cai, Qi GU, Xiang Wang, and An Zhang. Look back to reason forward: Revisitable memory for long-context LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026.

[37] Alan Baddeley. Working memory: Theories, models, and controversies. *Annual review of psychology*, 63(1):1–29, 2012.

[38] Norman E Spear. Forgetting as retrieval failure. *Animal memory*, pages 45–109, 1971.

[39] Laura L Eldridge, Stephen A Engel, Michael M Zeineh, Susan Y Bookheimer, and Barbara J Knowlton. A dissociation of encoding and retrieval processes in the human hippocampus. *Journal of Neuroscience*, 25(13):3280–3286, 2005.

[40] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *Arxiv:2412.15115*, 2024.

[41] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

[42] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, 2018.

[43] Donghee Lee, Jongmoo Choi, Jong-Hun Kim, Sam H Noh, Sang Lyul Min, Yookun Cho, and Chong Sang Kim. On the existence of a spectrum of policies that subsumes the least recently used (lru) and least frequently used (lfu) policies. In *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 134–143, 1999.

[44] Predrag R Jelenković and Ana Radovanović. Least-recently-used caching with dependent requests. *Theoretical computer science*, 326(1-3):293–327, 2004.

[45] Nimrod Megiddo and Dharmendra S Modha. Outperforming lru with an adaptive replacement cache algorithm. *Computer*, 37(4):58–65, 2004.

[46] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Igorevich Sorokin, Artyom Sorokin, and Mikhail Burtsev. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Appendix

Table of Contents

A Multi-Head Recurrent Memory Paradigm **14**
- A.1 Overview of MHM Pipeline . . . 14
- A.2 MHM-LRU: MHM Instantiation with Least Recently Updated Strategy . . . 14
- A.3 Theoretical Guarantee of MHM-LRU . . . 16
- A.4 MHM-Concur: MHM Instantiation with Concurrent Selection And Update . . . 17
- A.5 MHM-Relevance: MHM Instantiation with Model-Based Selection Strategy . . . 17

B Detailed Experiment Setting **17**
- B.1 Memory Setting . . . 17
- B.2 Runtime Setting . . . 18

C Additional Experiment Results **19**
- C.1 Ablation On Head Count Fixing Total Capacity . . . 19
- C.2 End-to-end Performance Results on Larger Models . . . 19
- C.3 Failure Case Analysis of MHM-LRU . . . 20

D Memory Head Diversity Analysis **20**
- D.1 Experiment Setup . . . 20
- D.2 Experiment Results . . . 21

E Broader Impact **21**

A Multi-Head Recurrent Memory Paradigm

A.1 Overview of MHM Pipeline

The pseudo-code of the MHM pipeline is shown in Algorithm 1.

Algorithm 1: Multi-Headed Memory For Long-Context Query

Input: Context C, query q, backbone LLM, number of heads H, chunk size L
Output: Response y
Initialize memory $m_0 \leftarrow \{m_0^1, \ldots, m_0^H | m_0^h = \emptyset, h \in [H]\}$;
Equally split C into context chunks $C_{\text{split}} \leftarrow [c_1, c_2, \ldots, c_T], T = \left\lceil \frac{|C|}{L} \right\rceil$.
for $t \leftarrow 1$ **to** T **do**
  Head selection: $h^\star \leftarrow \arg\max_{h' \in [H]} r_{\text{select}}(h' \mid m_{t-1}, q, c_t)$;
  Head update: $m_t^{h^\star} \leftarrow \text{LLM}(m_{t-1}, h^\star, q, c_t)$;
  Memory Overwrite: $m_t = \left(m_{t-1} \setminus \{m_{t-1}^{h^\star}\}\right) \cup \{m_t^{h^\star}\}$.
Generate response $y \leftarrow \pi_\theta(m_T, q)$;
return y;

A.2 MHM-LRU: MHM Instantiation with Least Recently Updated Strategy

The prompt template of MHM-LRU is shown in Figure 8.

Prompt Template: Stage-wise Memory Head Update and Final Answer

Prompt Template for Memory Update:

You are presented with a problem, a section of an article that may contain the answer to the problem, and previous memory. Please read the provided section carefully and update the memory *[*memory ID*]* with the new information that helps to answer the problem. Be sure to retain all relevant details from the previous memory while adding any new, useful information. Directly output the updated content without including the memory id. Be sure to note the source context location of the updated memory.

<problem>
[*Query* q]
</problem>

<memory_1>
[*Memory Head* m_{t-1}^1]
</memory_1>

<memory_2>
[*Memory Head* m_{t-1}^2]
</memory_2>

<memory_3>
[*Memory Head* m_{t-1}^3]
</memory_3>

<memory_4>
[*Memory Head* m_{t-1}^4]
</memory_4>

} [*Memory Head Set* $m_{t-1} = \{m_{t-1}^1, m_{t-1}^2, m_{t-1}^3, m_{t-1}^4\}$]

<section>
[*Context Chunk* c_t]
</section>

Updated [*memory ID*]:

Prompt Template for Final Answer:

You are presented with a problem and a previous memory containing multiple heads. Please answer the problem based on the previous memory and format your response as follows "Therefore, the answer is (insert answer here)".

<problem>
[*Query* q]
</problem>

[*Memory Head Set* $m_T = \{m_T^1, m_T^2, m_T^3, m_T^4\}$]

Your answer:

Figure 8: LLM prompt template used in MHM-LRU and MHM-Relevance. Italicized tokens in brackets denote runtime input parameters. It should be noticed that **the targeted memory head is selected and input as a parameter by the previous selection stage**.

A.3 Theoretical Guarantee of MHM-LRU

We formally establish the uniform head utilization guarantee of MHM-LRU claimed in Section 4. Let $n_h(t)$ denote the number of times head $h \in [H]$ has been selected for update by the end of step t. Note that $\sum_{h=1}^{H} n_h(t) = t$ for all $t \geq 0$.

Proposition 1 (Uniform Head Utilization)**.** *Under MHM-LRU with H heads all initialized identically, for all $t \geq 0$:*

$$\max_{h\in[H]} n_h(t) \;-\; \min_{h\in[H]} n_h(t) \;\leq\; 1. \tag{7}$$

Equivalently, every head satisfies

$$\left\lfloor \frac{t}{H} \right\rfloor \;\leq\; n_h(t) \;\leq\; \left\lceil \frac{t}{H} \right\rceil, \qquad \forall\, h \in [H]. \tag{8}$$

Proof. We proceed by induction on t.

Base case ($t = 0$): $n_h(0) = 0$ for all h, so $\max_h n_h(0) - \min_h n_h(0) = 0 \leq 1$.

Inductive step: Assume that at step t, $\max_h n_h(t) - \min_h n_h(t) \leq 1$. Let $n_{\min} \coloneqq \min_h n_h(t)$. By the inductive hypothesis, every head satisfies $n_h(t) \in \{n_{\min},\, n_{\min} + 1\}$.

MHM-LRU selects $h^\star = \arg\max_{h\in[H]} r_{\text{select}}(h \mid m_{t-1}, q, c_t) = \arg\min_h n_h(t)$, so $n_{h^\star}(t) = n_{\min}$. After the update:

$$n_h(t+1) = \begin{cases} n_{\min} + 1 & h = h^\star, \\ n_h(t) & h \neq h^\star. \end{cases}$$

We consider two sub-cases.

Sub-case A: All heads share count $n_{\min}$ at step t. After the update, $n_{h^\star}(t+1) = n_{\min} + 1$ and $n_h(t+1) = n_{\min}$ for all $h \neq h^\star$. Hence $\max_h n_h(t+1) - \min_h n_h(t+1) = 1 \leq 1$.

Sub-case B: At least one head other than $h^\star$ has count $n_{\min} + 1$ at step t.

- If there exists $h' \neq h^\star$ with $n_{h'}(t) = n_{\min}$, then after the update $\min_h n_h(t+1) = n_{\min}$ and $\max_h n_h(t+1) = n_{\min} + 1$, giving $\max - \min = 1 \leq 1$.
- If $h^\star$ is the *unique* head with count $n_{\min}$ at step t, then all other heads already hold count $n_{\min} + 1$. After the update every head has count $n_{\min} + 1$, so $\max - \min = 0 \leq 1$.

In all cases the invariant is preserved. By induction, the bound holds for all $t \geq 0$.

To verify the floor–ceiling form: from the invariant $n_h(t) \in \{n_{\min}, n_{\min} + 1\}$ and the identity $\sum_h n_h(t) = t$, we have $H \cdot n_{\min} \leq t \leq H \cdot (n_{\min} + 1)$, i.e., $n_{\min} = \lfloor t/H \rfloor$. Hence $\lfloor t/H \rfloor \leq n_h(t) \leq \lceil t/H \rceil$ for all h. □

Corollary 1 (Asymptotic Uniform Utilization)**.** *Under MHM-LRU, the per-head update fraction converges uniformly to $1/H$:*

$$\left| \frac{n_h(t)}{t} - \frac{1}{H} \right| \;\leq\; \frac{1}{t}, \qquad \forall\, h \in [H],\; t \geq 1. \tag{9}$$

In particular, $n_h(t)/t \to 1/H$ as $t \to \infty$, uniformly over all heads.

Proof. From Proposition 1, $\lfloor t/H \rfloor \leq n_h(t) \leq \lceil t/H \rceil$. Since $\lceil t/H \rceil - t/H \leq 1$ and $t/H - \lfloor t/H \rfloor \leq 1$, we have $|n_h(t) - t/H| \leq 1$, which gives $|n_h(t)/t - 1/H| \leq 1/t$. □

Remark. Proposition 1 implies that no head bears disproportionate overwrite pressure: across any H consecutive steps, every head is updated *exactly* once. This structural guarantee is the formal basis for the retention advantage of MHM-LRU observed in Section 5: by bounding the maximum number of overwrites any single head sustains, the probability that a previously captured answer survives to the final memory state is maximized uniformly across all heads. Crucially, this guarantee is *architectural* rather than behavioral. This guarantee holds regardless of the LLM backbone or task, and requires no additional inference overhead, since head selection reduces to a simple integer comparison.

A.4 MHM-Concur: MHM Instantiation with Concurrent Selection And Update

MHM-Concur is a vanilla instantiation of MHM. The motivation of MHM-Concur is that the memory agent should be capable of selecting and updating the memory head at the same time. As shown in Figure 9, in the memory update prompt template, the LLM is required to output the updated memory in a specific format that consists of the selected head ID and updated content. The final answer prompt template is the same as in Figure 8.

Prompt Template: Concurrent Memory Head Update

You are presented with a problem, a section of an article that may contain the answer to the problem, and previous memory. Please read the provided section carefully and update the memory with the new information that helps to answer the problem. Specifically, the memory consists of four memory heads: `memory_1, memory_2, memory_3, memory_4`. Please select one of the memory heads and update the memory. Be sure to diversify the use of different memory heads within `memory_1` to `memory_4` and avoid repeatedly selecting the same memory head for updates. Be sure to retain all relevant details from the previous memory while adding any new, useful information. Be sure to format your update as follows: "`<memory_id> (your updated memory content) </memory_id>`".

`<problem>`
[*Query* q]
`</problem>`

[*Memory Head Set* $m_{t-1} = \{m_{t-1}^1, m_{t-1}^2, m_{t-1}^3, m_{t-1}^4\}$]

`<section>`
[*Context Chunk* c_t]
`</section>`

Updated memory:

Figure 9: LLM prompt template for memory update in MHM-Concur. Italicized tokens in brackets denote runtime input parameters. Different from the stage-wise paradigm, MHM-Concur requires the LLM to **select and update memory heads at the same time**.

A.5 MHM-Relevance: MHM Instantiation with Model-Based Selection Strategy

MHM-Relevance is characterized by the stage-wise, model-driven, semantic-aware head selection — another important potential instantiation of MHM. Specifically, the selection criteria is explicitly regulated as selecting the head that is the most semantically similar to the section, while prioritizing selecting empty memory heads to avoid single-head memory pattern, as shown in Figure 10. The consequential memory update and final answer prompt templates are the same as MHM-LRU, as shown in Figure 8.

B Detailed Experiment Setting

B.1 Memory Setting

Memory Setting. The context chunk size is set to 5,000 tokens per chunk for long context splitting, consistent with the memory setting in [17]. The total memory capacity is set to 4,096 tokens for all recurrent memory agents, which requires setting the token generation limit for each method according to its number of heads.

- **MemAgent** (one single memory tag): The `max_new_tokens` is set to 4,096.
- **ReMem** (one current memory and one recalled memory tags): The `max_new_tokens` is set to 2,048, ensuring that the total capacity of the two memory heads equals 4,096.
- **MHM-LRU**: The number of heads is set to four, and the `max_new_tokens` is set to 1,024, ensuring that the total capacity of the four memory heads equals 4,096. At the beginning, when multiple

Prompt Template: Model-Based Relevance-Driven Selection

You are presented with a problem, a section of an article that may contain the answer to the problem, and previous memory. Please read the provided section carefully and select a memory head in `memory_1, memory_2, memory_3, memory_4` that is ***the most semantically similar to the section***. If there is an empty memory head, you should prioritize selecting it. Be sure to diversify the use of different memory heads within `memory_1` to `memory_4` and avoid repeatedly selecting the same memory head for updates.

<problem>
[*Query* q]
</problem>

<memory_1>
[*Memory Head* m_{t-1}^1]
</memory_1>

<memory_2>
[*Memory Head* m_{t-1}^2]
</memory_2>

<memory_3>
[*Memory Head* m_{t-1}^3]
</memory_3>

<memory_4>
[*Memory Head* m_{t-1}^4]
</memory_4>

<section>
[*Context Chunk* c_t]
</section>

Selected memory:

Figure 10: LLM prompt template for memory selection in MHM-Relevance. Italicized tokens in brackets denote runtime input parameters.

memory heads have the same highest age, MHM-LRU chooses the lowest available head index by default.

- **Ablation on # of heads for MHM-LRU**: The `max_new_tokens` is fixed to 1,024 regardless of the number of heads, ensuring that the agent's performance is not affected by the change of token generation limit.

B.2 Runtime Setting

Decoding Setting. In the LLM decoding setting, the `temperature` is set as 0.7. The `top_p` is set as 0.95. The timeout limit is set as 60 seconds per 1,024 output tokens. As a result, the timeout limit is 240 seconds for MemAgent, 120 seconds for ReMem, and 60 seconds for MHM-LRU. Each LLM calling has three tries in case of encountering an exception (e.g., a timeout exception). All experiments are run on a Linux server with 8× A100 GPUs. LLMs are deployed with vLLM 0.15.0.

Evaluation Setting. In evaluating the end-to-end performance (accuracy) of recurrent memory agents, we adopt substring exact matching. In the model's final response, we instruct the model to output the answer in the format "`Therefore, the answer is (insert answer here)`". The evaluation pipeline matches the output sentence starting with "`Therefore, the answer is` ", with the rest part is distinguished as the agent's prediction. The pipeline then executes substring exact matching between normalized ground truth and the agent's prediction to decide the accuracy score of the agent's final response.

C Additional Experiment Results

C.1 Ablation On Head Count Fixing Total Capacity

Table 6: Memory Retention Rate (MRR ↑ %) with respect to head count given fixed total memory capacity.

Length	Head Count				
	1	2	4	8	16
112K	57.89	89.77	86.34	94.51	100.00
224K	52.69	84.69	80.48	93.48	98.86
448K	46.24	78.72	79.66	89.25	91.40
896K	26.53	80.68	73.96	87.50	96.25

Table 7: Accuracy (Acc ↑ %) with respect to head count given fixed total memory capacity.

Length	Head Count				
	1	2	4	8	16
112K	42.19	54.69	50.26	58.59	61.72
224K	42.97	60.16	50.26	53.91	53.91
448K	33.33	50.00	48.70	60.94	59.38
896K	21.62	49.22	49.74	55.47	51.56

We performed an additional experiment in which the total memory capacity is fixed at 4096 tokens across all settings. Accordingly, the per-head capacity is 2048, 1024, 512, and 256 tokens for H = 2, 4, 8, and 16, respectively.

The new results (see Table 6 and Table 7) demonstrate that multi-head partitioning remains highly effective even when the total memory capacity is held constant. Memory retention exhibits a similar trend, increasing from 26.53% at 896K with H = 1 to 80.68% (H=2), 73.96% (H=4), 87.50% (H=8), and 96.25% (H=16), respectively.Interestingly, H = 16 achieves the highest memory retention rate (96.25%), whereas H = 8 achieves the highest end-to-end accuracy (55.47%). This suggests a trade-off between stronger overwrite protection (more heads) and the representational capacity of each individual head (smaller per-head memory).

C.2 End-to-end Performance Results on Larger Models

Table 8: Accuracy (%) results of larger models on RULER-HQA.

Model	Method	Length		
		224K	448K	896K
gpt-oss-120b	LLM Baseline	43.30	28.26	22.92
	MHM-LRU	**58.59**	**51.56**	**52.76**
Qwen2.5-32B-Instruct	LLM Baseline	25.00	13.28	7.81
	MHM-LRU	**60.94**	**57.81**	**51.56**

We added experiments with GPT-OSS-120B and Qwen2.5-32B-Instruct, using their maximum 128K context. The evaluations are based on RULER-HQA. As shown in Table 8, both base LLMs degrade sharply at larger context lengths, while MHM-LRU maintains stronger performance.

C.3 Failure Case Analysis of MHM-LRU

A failure case analysis of MHM-LRU is shown below. In this case, the agent previously captured the correct answer at step 74 of the memory iteration. However, the agent discarded this correct information at step 78.

Failure Case: BABILong-512K

Item ID: 2
Question: Where is Mary?
Correct answer: hallway
Predicted answer: Mary is currently traveling towards California with Wayland Norcross and Berrie McFarlane in Wayland's father's private ...
Number of chunks: 99
Answer present in at least one head: 6 out of 99 steps

The memory capture step is shown below. The bold text highlights the captured answer, while the underline highlights the relevant reasoning chain of the model:

Step 74: Correct Answer Captured

<memory_3>
Mary is studying medicine in Glasgow. Previously, she was known to be in the cottage preparing for a party with other characters and later journeyed to the office. Currently, no new location update for Mary has been provided other than her being in Glasgow to study medicine. During the recent section provided, which focuses on narratives about the Inquisition and the experiences of individuals subjected to its cruelty and injustice, there is a brief mention indicating that Mary moved to the **hallway**. However, this detail seems out of context with the surrounding narrative and likely pertains to a different individual's experience. Therefore, this movement to the **hallway** is not considered reliable for locating Mary, who is currently understood to be in Glasgow for her studies. Source context: Found within biographical narratives and historical testimonials centered around female medical students' experiences in Glasgow around the late 19th century and detailed discussions on the evolution of the British Constitution, intertwined with historical accounts of medieval governance and legal practices, and narratives of the Inquisition's activities.
</memory_3>

Although the model captures the "hallway" in its third memory head, the reasoning chain shows that this information is highly uncertain, as it is out of context, surrounded by distraction paragraphs. Therefore, the model abandoned the correct answer in Step 78, causing the failure outcome, as shown below:

Step 78: Correct Answer Discarded

<memory_3>
Mary is studying medicine in Glasgow. Previously, she was known to be in the cottage preparing for a party with other characters and later journeyed to the office. Currently, no new location update for Mary has been provided other than her being in Glasgow to study medicine. The recent section provided does not contain any direct information about Mary's current location. This passage includes narratives unrelated to Mary's situation and revolves around personal correspondences and family dynamics, particularly involving a character named Pauline writing letters and her interactions with family members regarding a relative named Hilary. Source context: Found within biographical narratives and historical testimonials centered around female medical students' experiences in Glasgow around the late 19th century and detailed discussions on the evolution of the British Constitution, intertwined with historical accounts of medieval governance and legal practices, and narratives of the Inquisition's activities.
</memory_3>

D Memory Head Diversity Analysis

D.1 Experiment Setup

An inherent research question for the multi-head recurrent memory agent is: ***Are memory heads differentiated or not?*** To answer this question, we propose a semantic distance-based analysis of

memory head diversity, demonstrating whether and to what extent memory heads semantically differ from one another.

Evaluation Metric. We measure the diversity of memory heads using the average cosine distance for memory states at each time step. Specifically, given a memory trajectory of a test item, denote $m_t = \{m_t^1, \ldots, m_t^H\}$ as the memory state of MHM at time step t. We first use a sentence transformer (`all-mpnet-base-v2` in the experiment) to obtain the semantic vector of each memory head, i.e.,

$$v_t^h = \text{SentenceTransformer}(m_t^h), h \in [H]. \tag{10}$$

Next, we calculate the average memory vector $\overline{v}_t = \frac{1}{H}\sum_{h=1}^{H} v_t^h$. We define the cosine distance between each memory head and the averaged memory by:

$$d_{\cos}(v_t^h, \overline{v}_t) = 1 - \cos(v_t^h, \overline{v}_t), \tag{11}$$

Finally, the **averaged cosine distance** $\overline{d}_t$ is defined by:

$$\overline{d}_t = \frac{1}{H}\sum_{h=1}^{H} d_{\cos}(v_t^h, \overline{v}_t) \tag{12}$$

The $\overline{d}_t$ is non-negative and measures the diversity of memory heads at time step t. When $\overline{d}_t = 0$, all memory heads are semantically identical. The higher the $\overline{d}_t$, the more diverse the memory heads are. For each memory trajectory, we visualize $\overline{d}_t$ with respect to time step t to see how memory diversity evolves as time goes by.

D.2 Experiment Results

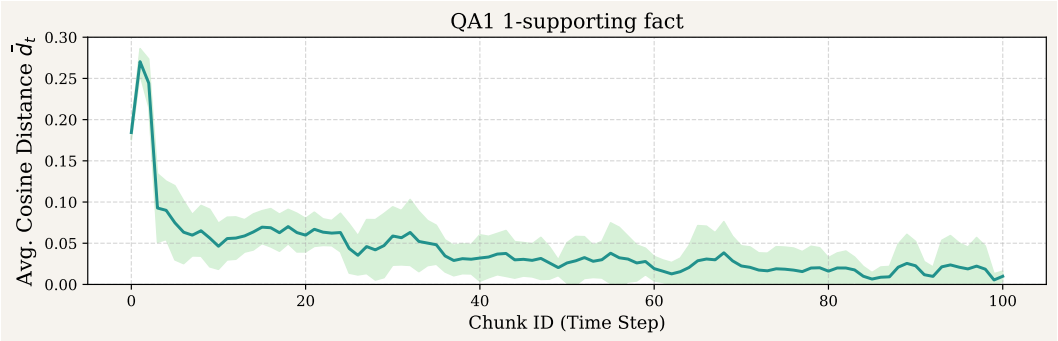
We evaluate memory diversity on three representative tasks in BABILong: the QA1 (1-supporting fact), QA4 (2-arg relations) and QA7 (Counting), choosing five trajectories with continuous item IDs for visualization for each task. Results are shown in Figure 11.

MHM Enables Diverse Memory Allocation. Figure 11 shows that MHM-LRU consistently produces non-trivial semantic diversity across memory heads throughout processing. Across all three tasks, $\overline{d}_t$ is positive at most time steps, confirming that heads spontaneously specialize without any explicit diversity objective. Diversity is highest in the initial steps when each head is first written, then gradually decreases and stabilizes at a lower plateau as MHM-LRU propagates task-relevant information redundantly across heads—the same emergent backup behavior documented in Section 5.5. This trajectory reflects a natural balance between early-stage coverage and late-stage retention robustness.

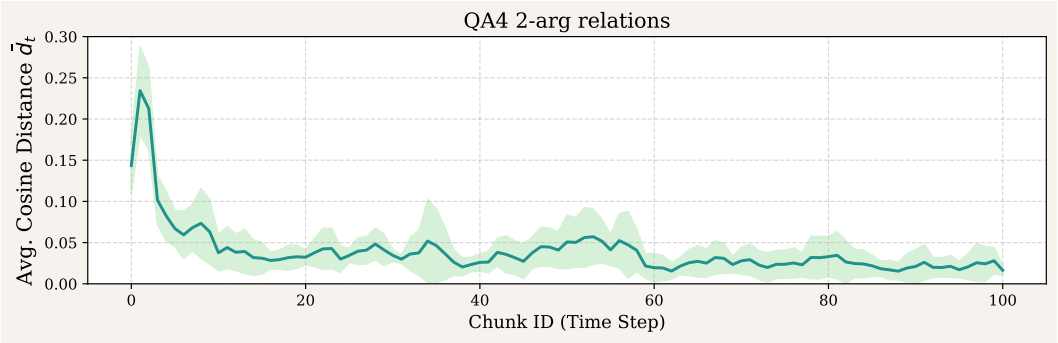
Diversity Varies In Different Tasks. The magnitude and duration of head diversity are modulated by task complexity. For QA1 (1-supporting fact), $\overline{d}_t$ collapses rapidly once the single target fact is captured, with heads converging to near-identical content by step 20–40. QA4 (2-arg relations) sustains higher diversity for longer. QA7 (Counting) exhibits the most persistent and fluctuating diversity, consistent with the incremental nature of counting where each new chunk may modify the running tally. This pattern aligns with the retention gains in Section 5.4: tasks requiring multi-step information accumulation benefit most from heads that remain semantically differentiated over longer processing horizons.

E Broader Impact

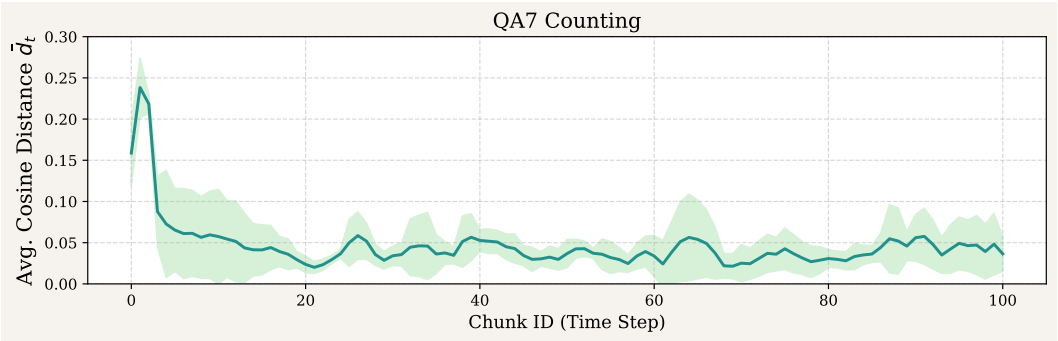
In this section, we discuss the broader impact of MHM not only on recurrent memory for long-context query, but more general research field of agentic memory in the future.



(a) QA1: 1-supporting fact.



(b) QA4: 2-arg relations.



(c) QA7: Counting.

Figure 11: Average memory head distance ($\overline{d}_t$) with respect to time steps. The higher the $\overline{d}_t$, the more semantically diverse the memory heads in the time step.

Towards Long-term and Heterogeneous Memory. Despite the success of MHM in single-turn long-context query, the current research focus of recurrent memory is still limited by query-specific, short-term memory. An inevitable following research question is: what a long-term memory should be, and how to maintain it. Long-term memory is valuable in the whole life cycle of an agent, containing a variety of heterogeneous memory information, and is query-agnostic. The multi-head memory has the potential to achieve this: by allowing each memory head to specialize in storing a distinct type of information, such as episodic experience, declarative knowledge, procedural skills, or relational context, MHM provides a natural architectural substrate for heterogeneous long-term memory. Unlike the monolithic single-head design, which is forced to compress all memory types into a single undifferentiated representation, the multi-head structure admits a principled partition of memory responsibilities across heads. This partitioning can be further guided by semantic-aware or type-aware selection strategies, enabling the agent to organize, retrieve, and update memory content in a structured and persistent manner across its entire operational lifetime.

Towards Self-Evolving Agent. We believe that a must-go path for memory agent is to achieve a realistic self-evolving agent—the agent can learn experience from past failure or success, and accumulate knowledge from observation and simulation. The multi-head memory essentially provides a technical path towards the self-evolving agent: by endowing agents with the ability to spontaneously and freely manage and utilize multiple memory heads, the agent can achieve self-evolvement of its experience through self-evolvement of memory internalization. In the long run, we envision that MHM-style architectures, when coupled with continual learning and reflection mechanisms, could enable agents to autonomously consolidate short-term observations into long-term behavioral priors, progressively refine their internal representations in response to environmental feedback, and ultimately exhibit sustained improvement over time without human intervention, which is a foundational step toward truly autonomous and adaptive AI systems.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contribution on recurrent memory agent, long context LLMs and their broader range of application.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the work are discussed in Section 6

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Full set of assumptions, correct and complete proof, are presented in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Experiment settings are demonstrated in Section 5 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Experiment data is publicly available online. Code will be released publicly upon publication.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Experiment settings are presented in Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the mean and standard deviation for evaluation results in main experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the type of computation workers, required memory and time of execution in the utility analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research confirm with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses the potential social impacts in Appendix E.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Data and models used in this paper are fully open-sourced and available on the Internet.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the data, code and model used in this paper are open-sourced and we rigorously obey the license and terms of use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets (mainly code) are well documented.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This research does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This research does not involve crowdsourcing nor reearch with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.