

InfMem: Learning System-2 Memory Control for Long-Context Agent

Xinyu Wang^{♣*} Mingze Li^{♣*} Peng Lu[♡] Xiao-Wen Chang[♣]
 Lifeng Shang[♣] Jinpeng Li[♣] Fei Mi[♣] Prasanna Parthasarathi[♣] Yufei Cui^{♣†}

♣ McGill University ♣ Noah’s Ark Lab ♡ Université de Montréal

Abstract

Reasoning over ultra-long documents requires synthesizing sparse evidence scattered across distant segments under strict memory constraints. While streaming agents enable scalable processing, their passive memory update strategy often fails to preserve low-salience *bridging evidence* required for multi-hop reasoning. We propose **InfMem**, a control-centric agent that instantiates System-2-style control via a PRETHINK–RETRIEVE–WRITE protocol. InfMem actively monitors evidence sufficiency, performs targeted in-document retrieval, and applies evidence-aware joint compression to update a bounded memory. To ensure reliable control, we introduce a practical SFT→RL training recipe that aligns retrieval, writing, and stopping decisions with end-task correctness. On ultra-long QA benchmarks from 32k to 1M tokens, InfMem consistently outperforms MemAgent across backbones. Specifically, InfMem improves average absolute accuracy by **+10.17**, **+11.84**, and **+8.23** points on Qwen3-1.7B, Qwen3-4B, and Qwen2.5-7B, respectively, while reducing inference time by $3.9\times$ on average (up to $5.1\times$) via adaptive early stopping.

Code is available at [code](#).

1 Introduction

Large Language models (LLMs) have achieved remarkable performance across a wide range of tasks (Achiam et al., 2023; Touvron et al., 2023; DeepSeek-AI, 2025). Despite these successes, they continue to exhibit limitations on tasks that require long-context abilities. In particular, long-document question answering increasingly demands inference over *extreme-length* inputs—ranging from multi-hundred-page legal contracts to codebases spanning hundreds of thousands of tokens—where correct answers hinge on a few *sparse, widely scattered* facts rather than the global gist (Shaham et al., 2022; An et al., 2024). Existing long-context LLM research largely focuses on improving the generic ability of models to operate over long inputs—either by (i) increasing the amount of text that can be ingested through architectural innovations (Liu et al., 2023; Gu & Dao, 2024; Yang et al., 2024b) or (ii) encouraging models to generalize beyond their pre-training context window (Press et al., 2022; Su et al., 2024; Peng et al., 2024a). and reduce length-extrapolation failures. While these methods mitigate the computation burden or preserve the basic language ability of LLM on extended inputs, the foundational LLMs still face challenges to make use of the relevant information in the extremely long contexts, as recent works show significant degradation when models must access correct information in the middle of long inputs (Liu et al., 2024b; Salvatore et al., 2025).

Instead of feeding an entire ultra-long document into an LLM directly, a practical alternative is streaming processing: the document is segmented into chunks read sequentially, and the model maintains a fixed-size bounded memory that is selectively overwritten at each step, yielding constant memory cost and linear-time computation (Packer et al., 2023; Yu et al.,

*Equal contribution.

†Corresponding authors: xinyu.wang5@mail.mcgill.ca, yufei.cui@huawei.com.

2025). While these methods improve efficiency, they introduce an unidirectional optimization bias. Specifically, every chunk is forced through the same fixed-size memory bottleneck, regardless of the complexity of the underlying question. This uniform treatment implicitly assumes that all segments of the document contribute equally to the final answer. However, long-document QA naturally contains a mixture of *high-complexity* and *low-complexity* queries. Some questions require multi-step reasoning over subtle cross-section dependencies, while others only demand direct retrieval of a single explicit fact. This mismatch creates a structural dilemma: to preserve information for complex questions, the memory must retain rich, high-fidelity representations of earlier content; yet to remain within the fixed memory budget, the system must heavily compress each chunk, inevitably discarding fine-grained details.

This structural dilemma exposes a deeper limitation of streaming-based long-context processing: the system cannot adapt its reasoning depth or memory preservation to the complexity of the question. Complex queries demand multi-step abstraction and the retention of subtle cross-section dependencies, yet a fixed-size memory with uniform compression treats all incoming information homogeneously. In the language of dual-process theory (Kahneman, 2011), such architectures are locked into fast, reflexive “System-1” heuristics where deliberate, state-dependent System-2 control is needed. While recent work on test-time scaling (Wei et al., 2022; Jaech et al., 2024; DeepSeek-AI, 2025) shows that LLMs can dynamically allocate computation at inference time, no existing streaming framework channels this capacity into principled memory control for long-context QA.

We address this gap with InfMem, a bounded-memory agent that replaces passive compression with an explicit, System-2-style control loop (Kahneman, 2011; Sumers et al., 2023), instantiated as a PRETHINK–RETRIEVE–WRITE protocol. At each streaming step, PRETHINK assesses whether the accumulated evidence suffices for the query; if not, RETRIEVE issues targeted queries against a global document index, and WRITE jointly compresses the retrieved passages, the current chunk, and the existing memory within a fixed budget. Once evidence is deemed sufficient, the loop stops early, protecting consolidated memory from redundant overwrites and curtailing computation. To make these control decisions reliable rather than heuristic, we combine supervised fine-tuning on reasoning-correct trajectories with reinforcement learning that aligns retrieval, writing, and stopping with end-task correctness and efficiency. On ultra-long QA benchmarks (up to 1M tokens), InfMem outperforms the state of the art by over **10 points** across three Qwen-series backbones while reducing inference latency by $3.9\times$ on average via adaptive early stopping.

2 Related Work

Long-Context Modeling and Efficiency. Frontier models now scale to million-token regimes (Qwen Team, 2025; Wan et al., 2025; Yang et al., 2025b; Wan et al., 2025), and efficient architectures such as linear attention and SSMs reduce the quadratic cost of self-attention (Gu & Dao, 2024; Yang et al., 2024b). Among these, refined gated linear attentions (Yang et al., 2025c; Lu et al., 2025b; Zhang et al., 2025) have emerged as promising variants that balance efficiency with expressiveness. However, both transformers and their efficient alternatives consistently suffer from length generalization issues when trained on short sequences and tested on long ones, requiring careful intervention either before or after training (Press et al., 2022; Peng et al., 2024b; Lu et al., 2025a). Limited capacity is a general challenge when processing long contexts for architectures with recurrent states or local windows, and incorporating additional modules to augment this capacity can improve performance on retrieval-heavy tasks (Wang et al., 2025b; Ma et al., 2026). Yet expanded capacity alone does not ensure effective reasoning: retrieval-heavy tasks still exhibit “lost-in-the-middle” degradation (Liu et al., 2024a; Weston & Sukhbaatar, 2023), and monolithic processing offers no explicit control over which evidence is retained or discarded. Integrating retrieval natively into the reasoning chain improves context fidelity (Wang et al., 2025a), but presumes the document fits within the context window and offers no mechanism for streaming consolidation under a memory budget.

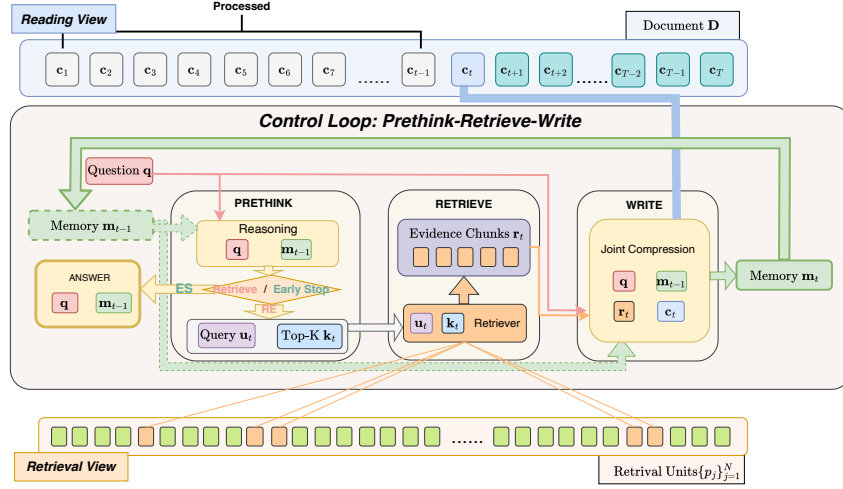


Figure 1: InfMem runs a **System-2 control loop** (PRETHINK–RETRIEVE–WRITE) over bounded memory. PRETHINK monitors sufficiency and triggers early stops; RETRIEVE fetches sparse evidence to bridge logical gaps; WRITE compresses evidence with the current stream to update memory, preserving fidelity under extreme context lengths.

Bounded-Memory Agents. Zhang et al. (2023) formulate LLMs as semi-parametric RL agents that learn to retrieve and update memory. MEM1 (Zhou et al., 2025), Memory-R1 (Yan et al., 2025), and MemGPT (Packer et al., 2023) introduce dedicated memory mechanisms but primarily target conversational settings (Maharana et al., 2024), prioritizing state tracking or persona consistency rather than evidence aggregation over a single document. Beyond textual memory, context can also be consolidated *parametrically*—by transforming it into weight updates (Cao et al., 2025) or via lifelong model editing (Wang et al., 2026)—which faces an analogous plasticity–stability dilemma between absorbing new evidence and preserving consolidated knowledge. For long-document QA, MemAgent (Yu et al., 2025) is the most directly relevant: it streams chunks sequentially and recurrently compresses each chunk with the existing memory. However, this fixed routine cannot assess memory sufficiency, revisit lost context, or terminate early.

3 InfMem: Streaming Inference with a Control Loop

In this section, we begin by formally defining the streaming long-context reasoning problem, then introduce InfMem, our inference protocol that augments such agents with a closed-loop control mechanism to dynamically manage what to retain, retrieve, or discard under a constrained memory budget.

3.1 Problem Formulation

Given a question q and a document D too long for a single context window, the goal is to produce answer $\hat{y}$ from evidence distributed throughout D . Following Yu et al. (2025), we segment D into T ordered chunks $\{c_t\}_{t=1}^T$ processed left-to-right, with the agent maintaining a bounded memory m_t under a fixed token budget $|m_t| \leq M$. After each chunk, the agent overwrites one memory entry to form m_{t+1} , keeping per-step context constant and end-to-end cost linear in T .

3.2 Control Loop: PreThink–Retrieve–Write with Early Stop

Algorithm 1 InfMem Control Loop

Require: question q ; streaming chunks $\{c_t\}_{t=1}^T$; retrieval units $\{p_j\}_{j=1}^N$; budget M

```

1:  $m_0 \leftarrow \emptyset$ 
2: for  $t = 1$  to  $T$  do
3:    $(a_t, u_t, k_t) \leftarrow \text{PRETHINK}(q, m_{t-1})$ 
4:   if  $a_t = \text{STOP}$  then
5:     break
6:   end if
7:    $r_t \leftarrow \text{RETRIEVE}(u_t, k_t; \{p_j\})$ 
8:    $m_t \leftarrow \text{WRITE}(q, m_{t-1}, c_t, r_t; M)$  s.t.  $|m_t| \leq M$ 
9: end for
10:  $\hat{y} \leftarrow \text{ANSWER}(q, m_*)$ 
11: return  $\hat{y}$ 

```

Overall Pipeline. **InfMem** boosts models with a control loop. At step t , it maintains a bounded memory m_{t-1} as its intermediate state and proceeds via three operations. First, **PRETHINK**, conditioned solely on (q, m_{t-1}) , **monitors** whether the accumulated evidence is sufficient to answer q . If so, the agent emits **STOP** and terminates early, avoiding unnecessary computation. Otherwise, it emits **RETRIEVE** together with a synthesized retrieval query and a predicted fetch count k_t , and invokes **RETRIEVE** to **seek** sparse evidence globally from the source document, producing a compact retrieved context r_t . Finally, **WRITE** **updates** the memory by jointly compressing the incoming chunk c_t , the retrieved context r_t , and the prior memory m_{t-1} under the fixed budget M , yielding m_t with $|m_t| \leq M$. We detail each operation below (see also Figure 1).

PRETHINK. **PRETHINK** is a state-dependent controller. Given (q, m_{t-1}) , it outputs a structured control record $c_t = (a_t, u_t, k_t)$ that specifies the step- t action:

- **ACTION** $a_t \in \{\text{"STOP"}, \text{"SEARCH"}\}$: whether the current memory is sufficient to answer q (stop) or additional in-document evidence is needed (retrieve);
- **QUERY** u_t (if $a_t = \text{"SEARCH"}$): a single dynamic query synthesized from (q, m_{t-1}) ;
- **TOPK** $k_t \in \{1, \dots, K_{\max}\}$ (if $a_t = \text{"SEARCH"}$): the number of retrieval units to fetch.

Together, (a_t, u_t, k_t) define the control decisions at step t : *whether to stop*, and if continuing, *what to search and how much to search*. Optionally, **PRETHINK** may also emit a brief natural-language rationale (e.g., missing evidence or subgoals) to improve interpretability and prompting, but these auxiliary fields do not affect execution beyond the induced u_t .

RETRIEVE. While the document is processed sequentially as coarse streaming chunks, we strictly distinguish the *reading view* from the *retrieval view*. We pre-construct a finer-grained set of *retrieval units* $\{p_j\}_{j=1}^N$ (e.g., paragraphs) from the same document. Unlike the coarse streaming chunks, these units are compact and globally indexed. When triggered by **PRETHINK**, **InfMem** can search any part of the document (past or future) to retrieve the top- k_t units and summarize them into a concise context r_t , while preserving the coarse-grained reading flow. If $a_t = \text{"SEARCH"}$, **InfMem** retrieves top- k_t relevant retrieval units from the same document (no external corpus) and concatenates them into a compact context:

$$\begin{aligned}
 P_t &\leftarrow \text{RETRIEVE}(u_t, k_t; \{p_1, \dots, p_N\}), \\
 r_t &\leftarrow \text{Concat}(P_t),
 \end{aligned} \tag{1}$$

with separators and (optionally) unit identifiers to preserve provenance.

WRITE. If $a_t = \text{"SEARCH"}$, **InfMem** overwrites the memory with a bounded new state:

$$m_t \leftarrow \text{WRITE}(q, m_{t-1}, c_t, r_t; M), \quad \text{s.t. } |m_t| \leq M. \tag{2}$$

WRITE has access to (q, m_{t-1}, c_t) as well as the retrieved evidence r_t , and then performs evidence-aware composition: it connects the retrieved support r_t with the newly observed content in c_t in order to identify and encode the composition-critical facts and the bridging links into a bounded updated memory. We refer this overwrite update to as **joint compression**, where the retrieval is used *for writing* to shape the memory update.

EARLY STOP. If $a_t = \text{"STOP"}$ at a step, the agent halts the retrieval and the memory updates: it directly produces the final answer using the current memory. Otherwise, it continues until the end of the chunk stream ($t = T$). After termination (early stopping or reaching the end-of-sequence), **InfMem** generates:

$$\hat{y} \leftarrow \text{ANSWER}(q, m_\star),$$

where $m_\star$ denotes the final memory state at termination.

4 Training InfMem

The PRETHINK–RETRIEVE–WRITE protocol poses training challenges absent in standard language modeling: the model must produce heterogeneous outputs (retrieval decisions, free-form queries, and compressed memory) whose quality is only revealed at the end of a long rollout, with no ground-truth labels for intermediate decisions. This motivates a two-stage recipe. **Stage 1 (SFT)** teaches the model to produce protocol-valid outputs—well-formed function calls, schema-consistent memory updates, and formatted answers. **Stage 2 (RL)** then optimizes the control policy—when to stop, what to retrieve, and how to compress—via outcome-based rewards propagated through the full trajectory.

4.1 SFT Warmup via Supervised Distillation

We distill a strong teacher (Qwen3-32B) into a smaller student using trajectories from the PRETHINK–RETRIEVE–WRITE loop. For each question, the teacher executes the full protocol, optionally guided by golden question decompositions (e.g., for MuSiQue) to elicit cleaner retrieval traces, in the spirit of evidence-grounded trajectory synthesis (Li et al., 2026). Crucially, these hints are given *only to the teacher*; the student never observes decompositions, gold document IDs, or any teacher-side annotations. The subsequent RL stage operates entirely without such hints yet yields substantial additional gains (Table 6), confirming that the learned control policy—not teacher supervision—drives the final performance.

Data filtering and training objective. We construct a warmup set from HotpotQA, SQuAD, and MuSiQue (Appendix A.2.1). Only trajectories whose final answer is correct ($\text{EM}(\hat{y}, y) = 1$) are retained; string and regex filters remove any ground-truth leakage. Each retained rollout is serialized into a protocol-formatted dialogue τ , and the student is trained with masked next-token prediction on *agent response tokens* only (masking all system/user/prompt tokens):

$$\mathcal{L}_{\text{SFT}} = - \sum_{\tau \in \mathcal{D}_{\text{SFT}}} \sum_{i \in \mathcal{Y}(\tau)} \log \pi_\theta(y_i \mid \text{prefix}_i(\tau)), \quad (3)$$

where $\mathcal{Y}(\tau)$ indexes the response tokens and $\text{prefix}_i(\tau)$ denotes all preceding tokens. Gradients propagate through all realized steps, jointly supervising the control records, bounded memory updates, and final answers. In practice, SFT primarily teaches the *mechanics* of the protocol; the control policy is refined in the subsequent RL stage (§4.2).

4.2 RL Alignment with Reward Design

While SFT ensures protocol-compliant execution, it does not optimize the *control policy* itself—when to stop, what to retrieve, and how to compress under the overwrite budget. We apply RL to align these decisions with end-task success and efficiency. A central design challenge is that InfMem’s control loop produces long-horizon trajectories with *delayed, sparse* feedback: only the final answer is verifiable, yet the decisive actions—a well-targeted retrieval query, a faithful memory compression, or a timely stop—may occur many steps earlier. Our reward design addresses this by combining four complementary signals that together supervise task correctness, stopping efficiency, and protocol integrity.

Task correctness reward. The primary signal is a binary ground-truth reward on the final predicted answer:

$$R_{\text{gt}}(\hat{y}, y) = \mathbf{1}\{\text{equiv}(\hat{y}, y)\}, \quad (4)$$

where $\text{equiv}(\cdot, \cdot)$ follows the official benchmark evaluation protocol (e.g., exact-match normalization). This reward drives the agent to retrieve the right evidence and compress it faithfully, but alone provides no signal for *when* to stop.

Early-stop shaping. Continuing after the memory already suffices wastes compute and risks displacing consolidated evidence via redundant overwrites. We introduce a shaping term that rewards stopping soon after the memory first becomes sufficient. Let t_{first} be the earliest step at which the current memory alone yields a correct answer (EM=1 under the official evaluation protocol), and let t_{stop} be the agent’s actual stopping step. With $d = t_{\text{stop}} - t_{\text{first}}$:

$$R_{\text{early}} = \gamma^{d-1}, \quad \gamma \in (0, 1), \quad (5)$$

so $R_{\text{early}} = 1$ for immediate stopping after sufficiency, decaying exponentially with delay. The sufficiency check uses the same EM protocol as final evaluation; while it may occasionally misjudge on hard multi-hop cases, the group-level advantage normalization during optimization averages over G rollouts, smoothing individual noisy signals.

Protocol-soundness verifiers. Long-horizon agentic rollouts are fragile: a single malformed function call or truncated memory update can corrupt all downstream steps. We add two binary rollout-level verifiers to keep RL exploration within protocol-valid regions: (i) $R_{\text{call}} = 1$ iff all function calls are well-formed and parsable; (ii) $R_{\text{mem}} = 1$ iff every memory-update step outputs a complete, non-truncated UPDATEDMEMORY field within the fixed budget. The four signals are combined into a single outcome reward per rollout:

$$R = \sum_w \alpha_w R_w, \quad w \in \{\text{gt}, \text{early}, \text{call}, \text{mem}\}, \quad (6)$$

where all coefficients α_w are specified in §5.1.

Optimization. We optimize the composite reward R using group relative policy optimization (GRPO). Each rollout spans multiple memory-update steps and a final answering step, with R shared across all preceding steps for long-horizon credit assignment. For each query we sample G rollouts, compute mean-centered advantages $\hat{A}_i = R_i - \bar{R}$, and optimize:

$$J(\theta) = \mathbb{E}_{i,t} \left[\min(r_{i,t}(\theta) \hat{A}_i, \text{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_i) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}) \right], \quad (7)$$

where $r_{i,t}(\theta) = \pi_\theta(a_{i,t} | s_{i,t}) / \pi_{\theta_{\text{old}}}(a_{i,t} | s_{i,t})$.

5 Experiments and Results

5.1 Experimental Setup

Datasets and backbones. We utilize four datasets spanning a spectrum of reasoning demands: SQuAD (Rajpurkar et al., 2016), HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022). These corpora form the basis for constructing our synthetic long-context training data and evaluation benchmarks, as detailed in §A.2. In addition, we report results on LongBench (Bai et al., 2024), a standardized long-context benchmark suite that evaluates LLMs under unified prompts and consistent scoring across diverse long-document QA tasks as complements to our controlled settings. We evaluate InfMem on Qwen3-1.7B, Qwen3-4B (Yang et al., 2025a), and Qwen2.5-7B-Instruct (Yang et al., 2024a) as base policies π_θ for both SFT and RL stages.

Training Data and Settings. We pack synthetic PRETHINK–RETRIEVE–WRITE trajectories from HotpotQA, SQuAD, and MuSiQue (§A.2.1) into 32k-token sequences for SFT, using Qwen3-32B as the distillation teacher. RL training uses a long-context variant of HotpotQA (§A.2.2) for dense multi-hop signals. For Stage 1 (SFT), we use a cosine-decayed learning rate of 4.0×10^{-5} with batch size 256; Qwen3-1.7B and 4B are trained for 1 epoch, while Qwen2.5-7B-Instruct is trained for 4 epochs. For Stage 2 (RL), we apply GRPO ($G=4$ rollouts, temperature 1.0, top- $p=1.0$, $\beta=0.001$) with a constant learning rate of 1×10^{-6} and batch size 128 (mini-batch 8). Detailed hyperparameters are provided in §5.1.

Table 1: Performance on synthesized RULER-style benchmarks across Qwen3-4B and Qwen2.5-7B under increasing context lengths. To ensure robust baselines, RESP directly uses Qwen3-32B, while Mem- α pairs its context manager with Qwen3-32B for question answering. Overall, MemAgent and InfMem provide consistent train-free gains over baselines (including YaRN and RAG), with RL further amplifying these improvements.

Metric	Other approach		Qwen3-4B						Qwen2.5-7B					
	RESP	mem- α	Framework				+RL		Framework				+RL	
			YaRN	RAG	MemAgent	InfMem	MemAgent	InfMem	YaRN	RAG	MemAgent	InfMem	MemAgent	InfMem
avg	30.29	37.42	25.45	26.05	<u>43.61</u>	50.25	54.56	66.40	21.41	19.77	<u>37.06</u>	47.73	52.07	60.30
HQA														
28k	44.22	46.18	50.77	48.46	<u>52.55</u>	59.73	71.18	71.44	35.70	33.51	<u>44.96</u>	45.70	65.58	59.20
56k	45.59	42.42	42.07	43.69	<u>51.27</u>	58.69	66.21	68.73	31.74	29.87	<u>45.93</u>	48.56	62.88	62.23
112k	41.10	45.62	35.19	42.52	<u>44.02</u>	51.33	62.42	71.24	25.42	31.45	<u>42.76</u>	47.98	61.55	57.75
224k	25.80	40.02	10.96	24.11	<u>44.68</u>	47.82	59.12	67.42	13.93	16.94	<u>34.77</u>	49.65	59.95	60.55
448k	20.43	35.99	8.34	13.27	<u>40.47</u>	51.71	58.84	67.75	9.23	8.61	<u>33.07</u>	46.70	57.09	62.34
896k	16.72	35.84	5.26	3.73	<u>40.03</u>	49.07	51.70	66.13	3.91	2.39	<u>34.47</u>	42.60	58.39	57.51
SQuAD														
32k	46.08	45.89	48.70	<u>55.66</u>	53.82	65.75	69.49	65.31	34.36	36.80	<u>45.02</u>	55.77	61.95	61.70
64k	48.29	45.36	39.80	49.91	<u>54.73</u>	61.07	69.84	66.42	31.55	33.55	<u>47.06</u>	53.98	57.94	64.19
128k	40.36	45.10	36.79	45.05	<u>51.80</u>	64.17	72.96	66.05	29.44	27.93	<u>49.15</u>	54.23	58.26	58.82
256k	38.03	40.86	24.89	35.76	<u>46.23</u>	59.23	71.24	63.53	27.50	22.15	<u>41.83</u>	53.03	53.23	61.71
512k	31.62	52.68	34.93	26.36	<u>51.62</u>	64.99	77.21	78.12	20.23	20.70	<u>50.92</u>	63.08	69.85	69.27
1M	17.50	49.72	9.63	5.26	<u>48.91</u>	59.38	77.74	73.81	4.59	2.57	<u>44.97</u>	55.99	68.63	67.71
MuSiQue														
32k	28.45	29.17	19.65	19.84	<u>29.02</u>	41.35	41.79	56.58	19.93	18.37	<u>31.45</u>	37.09	36.67	46.27
64k	31.52	25.33	14.00	11.94	<u>34.03</u>	38.06	41.55	57.19	17.27	14.13	<u>26.29</u>	36.73	32.82	46.05
128k	26.78	23.08	9.48	15.78	<u>28.23</u>	35.31	36.62	55.62	10.61	10.40	<u>20.71</u>	41.33	37.79	48.13
256k	24.73	30.77	14.52	13.75	<u>25.50</u>	38.04	43.04	61.39	12.79	10.86	<u>25.52</u>	38.79	44.91	57.49
512k	22.23	31.21	7.48	7.97	<u>32.93</u>	31.57	35.64	59.59	12.94	10.12	<u>21.49</u>	40.14	35.77	55.26
1M	17.70	30.45	8.30	3.80	<u>25.62</u>	34.20	35.91	56.86	3.15	2.55	<u>21.77</u>	41.49	38.40	58.57
2Wiki														
32k	37.19	37.25	49.71	39.58	55.62	<u>54.70</u>	56.43	70.66	37.41	39.20	<u>44.71</u>	44.52	49.57	68.78
64k	32.52	31.54	40.86	28.34	<u>47.56</u>	49.88	48.55	74.84	42.00	37.38	<u>40.06</u>	49.53	51.68	64.80
128k	34.96	36.91	39.72	30.47	50.27	<u>47.40</u>	46.18	70.46	33.51	29.75	<u>46.31</u>	50.60	49.66	61.88
256k	22.86	32.26	20.39	20.02	<u>43.99</u>	46.20	41.42	66.62	21.67	13.96	<u>34.85</u>	48.15	47.73	63.55
512k	15.12	36.73	20.16	22.52	<u>48.23</u>	50.81	39.09	71.54	22.77	15.66	<u>32.93</u>	54.74	46.31	65.19
1M	17.13	27.77	19.12	17.30	<u>45.45</u>	45.55	35.18	66.39	12.15	5.60	<u>28.49</u>	50.62	43.18	68.20

5.2 Results on Various Input Lengths up to 1M Tokens

Table 1 reports results on synthesized long-context QA benchmarks evaluated across Qwen3-4B and Qwen2.5-7B. We compare our methods against standard long-context baselines (YaRN, RAG) and strong iterative or memory-construction frameworks like RESP and Mem- α (see Appendix B for full reproduction details). Two further retrieval-based baselines evaluated under a matched in-document corpus—Search-R1 (Jin et al., 2025), an RL-trained retrieval-decision policy, and Self-RAG (Asai et al., 2024), in both a prompted train-free variant and a checkpoint we train ourselves—are reported in Appendix F.2; InfMem outperforms both even when they use substantially larger backbones.

Models equipped with the length generalization method (YaRN (Peng et al., 2024b)) or conventional retrieval mechanism (RAG (Lewis et al., 2020)) degrade sharply beyond 128k tokens, often collapsing to single digits at 1M. iterative RAG system (RESP (Jiang et al., 2025)) and recent bounded memory agent method (Mem- α (Wang et al., 2025c)) mitigate this decline but still underperform both MemAgent and InfMem on average. InfMem consistently achieves the strongest results across all settings.

This performance advantage becomes particularly pronounced on complex multi-hop reasoning tasks where evidence is scattered. For instance, on the MuSiQue benchmark at 1M tokens (using Qwen3-4B+RL), InfMem achieves a score of 56.86 compared to MemAgent (Yu et al., 2025)’s 35.91. Similarly, on 2WikiMultiHopQA at 1M tokens, InfMem reaches 66.39 versus 35.18. On single-hop tasks like SQuAD, the performance gap narrows slightly, as MemAgent’s fixed, reactive compression is generally sufficient for direct fact extraction. However, for multi-hop queries, MemAgent’s recurrent architecture is highly susceptible to gradual information loss over extended horizons. In contrast, InfMem leverages explicit rea-

Table 2: **Performance comparison on the LongBench QA benchmark.** We evaluate Qwen series models across five QA datasets. The colored bars indicate the absolute performance gain (green) or loss (red) compared to the YaRN baseline. **InfMem** and its RL variant (highlighted in gray) consistently outperform other methods across all model scales.

Model	Method	LongBench QA					avg
		NQA	HQA	2Wiki	Qasper	Musique	
Qwen3-1.7B	YaRN	17.09	33.87	50.32	37.91	23.86	32.61
	MemAgent	15.04 -2.05	41.68 +7.81	34.04 -16.28	30.94 -6.97	19.84 -4.02	28.31 -4.30
	InfMem	20.25 +3.16	48.73 +14.86	54.05 +3.73	33.91 -4.00	28.40 +4.54	37.07 +4.46
	MemAgent +RL	19.23 +2.14	50.22 +16.35	47.58 -2.74	35.48 -2.43	30.35 +6.49	35.90 +3.29
	InfMem +RL	19.23 +2.14	59.28 +25.41	55.02 +4.70	33.19 -4.72	40.98 +17.12	41.54 +8.93
Qwen3-4B	YaRN	21.46	53.20	50.31	40.14	32.18	39.46
	MemAgent	20.22 -1.24	57.67 +4.47	59.09 +8.78	33.52 -6.62	32.12 -0.06	40.52 +1.06
	InfMem	23.27 +1.81	60.96 +7.76	69.66 +19.35	35.14 -5.00	44.19 +12.01	46.64 +7.18
	MemAgent +RL	20.74 -0.72	63.80 +10.60	67.83 +17.52	41.02 +0.88	42.14 +9.96	47.11 +7.65
	InfMem +RL	20.77 -0.69	65.14 +11.94	74.76 +24.45	40.74 +0.60	53.22 +21.04	50.93 +11.47
Qwen2.5-7B	YaRN	16.12	42.92	40.55	28.84	19.28	29.54
	MemAgent	19.86 +3.74	53.23 +10.31	55.40 +14.85	31.63 +2.79	36.52 +17.24	39.33 +9.79
	InfMem	19.76 +3.64	52.95 +10.03	48.78 +8.23	31.09 +2.25	31.69 +12.41	36.85 +7.31
	MemAgent +RL	19.47 +3.35	56.17 +13.25	57.66 +17.11	35.52 +6.68	31.23 +11.95	40.01 +10.47
	InfMem +RL	20.43 +4.31	60.34 +17.42	65.19 +24.64	35.68 +6.84	50.66 +31.38	46.46 +16.92

soning and decision-making capabilities to dynamically decompose complex questions into manageable sub-problems. Finally, the proposed SFT→RL training recipe further amplifies these structural advantages by optimizing the agent’s sequential decision-making and adaptive stopping policies. Consequently, RL-InfMem establishes a decisive lead, outperforming RL-MemAgent by an average margin of over 10 points across both evaluated backbones.

5.3 Results on Standard LongBench

Crucially, these gains are not confined to our synthesized ultra-long setting. As shown in Table 2, the performance improvements transfer to LongBench QA, where contexts are shorter but information density is higher—placing greater emphasis on evidence analysis and selection rather than on maintaining memory over long horizons (see detailed discussion in Section E.2).

Across backbone architectures, InfMem consistently outperforms MemAgent in both train-free and RL-enhanced configurations. Notably, while InfMem already leads without reinforcement learning, incorporating RL further widens the gap over strong baselines such as YaRN. Overall, these results indicate that InfMem strengthens not only robustness at extreme lengths (up to 1M tokens) but also the quality of reasoning-oriented evidence management on standard long-context QA benchmarks, demonstrating versatility across diverse task conditions.

6 Analysis and Discussion

6.1 Robustness Analysis across Context Length

We further compare our method with strong LLMs supporting long-context, including Qwen3-Next-80B-A3B-instruct (Qwen Team, 2025), QwenLong-L1-32B (Wan et al., 2025) and Deepseek-R1-Distill-Qwen-32B (DeepSeek-AI, 2025). The top rows of Figure 2 summarizes long-context scaling on Qwen3-4B up to 1M tokens. Despite extended context windows, accuracy often deteriorates in the ultra-long regime where evidence is sparse and separated by long gaps. InfMem remains substantially more stable beyond 128K tokens, and its advantage grows with length, especially on multi-hop datasets. We attribute this to sufficiency-aware control over retrieval and memory writing, which mitigates long-horizon

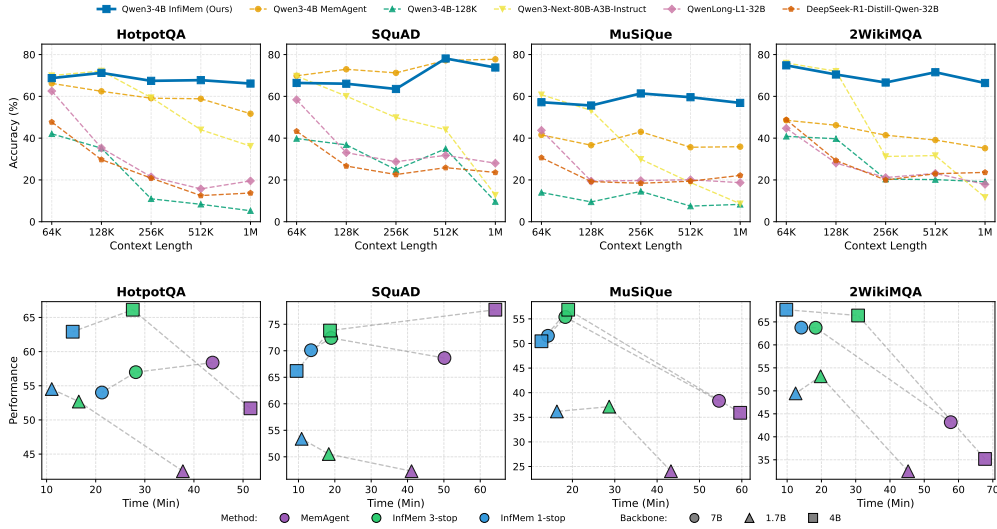


Figure 2: **Robustness and Inference Efficiency of InfMem on Synthesized QA Benchmarks.** **Top row:** Performance scaling of Qwen3-4B across context lengths from 64K to 1M tokens. InfMem demonstrates remarkable robustness, maintaining consistent accuracy without performance degradation as context length increases. **Bottom row:** Inference efficiency versus QA performance at 1M context length. InfMem exhibits exceptional proficiency in long-range multi-hop reasoning, preserving high-fidelity performance without the computational overhead typically associated with extreme sequence lengths.

drift from repeated compression and enables targeted recovery of missing bridging facts before updating memory. Qualitative case studies are provided in Section C.

6.2 Inference Efficiency Analysis

Table 3: **Decode (generated) tokens per query** at a short (length 800, ctx $\approx 112k$) and a long (length 6400, ctx $\approx 896k$) setting. Among the methods that scale to ultra-long context, InfMem is far cheaper than MemAgent and grows much more slowly ($\approx 5\times$ vs. $\approx 11\times$ from 800 to 6400).

Method	Backbone	@800	@6400
Mem- α	4B	15,994	114,992
MemAgent	4B	10,697	117,418
InfMem (ours)	4B	6,728	33,085

The bottom row of Figure 2 illustrates the efficiency-quality trade-off on 1M-token tasks. Across Qwen3-4B and Qwen2.5-7B, InfMem outperforms MemAgent on both axes: it improves accuracy by 11.80, 11.67, and 7.73 points, while reducing latency by 5.1x, 3.3x, and 3.3x. The conservative 3-stop policy further gains 2.76 points yet remains under half the runtime of MemAgent. These results confirm that InfMem reliably stops upon collecting sufficient evidence, avoiding redundant steps and establishing a superior efficiency-accuracy frontier. A more detailed analysis of early stopping behaviors and their direct impact on latency is provided in Appendix D.2.

Token cost. Since wall-clock time is partly shaped by serving-level batching, we additionally report the decode cost of each method. Table 3 lists generated tokens per query at $\approx 112k$ and $\approx 896k$ context. Among the methods that remain functional at ultra-long inputs, InfMem is the cheapest: at $\approx 896k$ it decodes 33.1K tokens per query against 117.4K for MemAgent (0.28 $\times$) and 115.0K for Mem- α , and its cost grows only $\approx 5\times$ from 112k to 896k

while MemAgent’s grows $\approx 11\times$. A per-length breakdown of tokens and latency, together with retrieval and index-storage accounting, is given in Appendix F.3.

6.3 Further Analysis and Ablation Study

Beyond the main results, we provide comprehensive ablation and further studies in Appendix D. These include retrieval chunk size selection, ablation on the thinking mode, and an analysis of memory retention. Appendix F additionally reports a zero-shot transfer of the InfMem protocol to a non-Qwen backbone (S.F.1), where the marginal gain on multi-hop QA is nearly identical to the one observed on Qwen3-4B.

7 Conclusion

In this work, we present InfMem, a cognitive agent designed to resolve the fidelity dilemma in ultra-long context reasoning. By integrating structured evidence management with a robust SFT→RL training pipeline, InfMem excels in long-horizon search and retrieval. Empirical evaluations on 1M-token benchmarks demonstrate that InfMem outperforms the state-of-the-art MemAgent with double-digit accuracy improvements across various Qwen models, while simultaneously reducing latency by $3.9\times$ via inference early stopping. Our findings suggest that as context windows scale, the primary bottleneck shifts from raw memory capacity to cognitive control: the ability to effectively discern and “know what is known”.

Ethics Statement

This work proposes a method for improving long-document question answering under bounded compute and memory budgets. The approach does not introduce new datasets, collect personal data, or target high-risk application domains. Potential benefits include improved efficiency and reliability in document analysis tasks such as technical review and knowledge synthesis. In this work, LLMs were used solely for grammar checking and textual correcting, consistent with accepted usage guidelines.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. L-eval: Instituting standardized evaluation for long context language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14388–14411, 2024.
- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=hSyW5go0v8>.
- Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL 2024, Bangkok, Thailand, August 11-16, 2024, pp. 3119–3137. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.172. URL <https://doi.org/10.18653/v1/2024.acl-long.172>.
- Bowen Cao, Deng Cai, and Wai Lam. InfiniteICL: Breaking the limit of context window size via long short-term memory transformation. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

- DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL <https://doi.org/10.48550/arXiv.2501.12948>.
- DeepSeek-AI. Deepseek-r1-distill-qwen model card. Hugging Face model repository, 2025. URL <https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B>. Accessed 2026-01-28.
- Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Núria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pp. 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/V1/2020.COLING-MAIN.580. URL <https://doi.org/10.18653/v1/2020.coling-main.580>.
- Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- Zhouyu Jiang, Mengshu Sun, Lei Liang, and Zhiqiang Zhang. Retrieve, summarize, plan: Advancing multi-hop question answering with an iterative approach, 2025. URL <https://arxiv.org/abs/2407.13101>.
- Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Serkan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.
- Daniel Kahneman. *Thinking, fast and slow*. macmillan, 2011.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>.
- Muzhi Li, Jinhu Qi, Yihong Wu, Minghao Zhao, Liheng Ma, Yifan Li, Xinyu Wang, Zhenghan Tai, Zixing Song, Yingxue Zhang, Ho-fung Leung, and Irwin King. From evidence to trajectory: Abductive reasoning path synthesis for retrieval-augmented generation agents development. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2026.
- Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranajpe, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024a.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranajpe, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173, 2024b. doi: 10.1162/TACL\A\00638. URL <https://doi.org/10.1162/tacl.a.00638>.
- Peng Lu, Jerry Huang, QIUHAO Zeng, Xinyu Wang, Boxing Chen, Philippe Langlais, and Yufei Cui. Mamba modulation: On the length generalization of mamba models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL <https://openreview.net/forum?id=QEU047bE8p>.

- Peng Lu, Ivan Kobyzev, Mehdi Rezagholizadeh, Boxing Chen, and Philippe Langlais. ReGLA: Refining gated linear attention. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2884–2898, Albuquerque, New Mexico, April 2025b. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.147. URL <https://aclanthology.org/2025.naacl-long.147/>.
- Linrui Ma, Chun Hei Lo, Xinyu Wang, Peng Lu, Xihao Yuan, Hanting Chen, Kai Han, Xinghao Chen, Chengjun Zhan, Hanlin xu, Yichun Yin, Lifeng Shang, Feng Wen, Boxing Chen, and Yufei Cui. MATCH: Modulating attention via in-context retrieval for long-context transformers. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15165–15179, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.692. URL <https://aclanthology.org/2026.acl-long.692/>.
- Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.
- Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. doi: 10.48550/ARXIV.2310.08560. URL <https://doi.org/10.48550/arXiv.2310.08560>.
- Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL <https://openreview.net/forum?id=wHBfxhZu1u>.
- Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024b. URL <https://openreview.net/forum?id=wHBfxhZu1u>.
- Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=R8sQPpGCv0>.
- Qwen Team. Qwen3-next: Hybrid attention and sparse moe (model release notes). Qwen blog, 2025. URL <https://qwen.ai/blog?from=research.latest-advancements-list&id=4074cca80393150c248e508aa62983f9cb7d27cd>. Accessed 2026-01-28.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Jian Su, Xavier Carreras, and Kevin Duh (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 2383–2392. The Association for Computational Linguistics, 2016. doi: 10.18653/V1/D16-1264. URL <https://doi.org/10.18653/v1/d16-1264>.
- Nikolaus Salvatore, Hao Wang, and Qiong Zhang. Lost in the middle: An emergent property from information retrieval demands in llms. *CoRR*, abs/2510.10276, 2025. doi: 10.48550/ARXIV.2510.10276. URL <https://doi.org/10.48550/arXiv.2510.10276>.
- Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, et al. Scrolls: Standardized comparison over long language sequences. *arXiv preprint arXiv:2201.03533*, 2022.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.
- Theodore Sumers, Shunyu Yao, Karthik R Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2023.

- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554, 2022. doi: 10.1162/TACL\A\00475. URL https://doi.org/10.1162/tacl-a_00475.
- Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. Qwenlong-11: Towards long-context large reasoning models with reinforcement learning. *CoRR*, abs/2505.17667, 2025. doi: 10.48550/ARXIV.2505.17667. URL <https://doi.org/10.48550/arXiv.2505.17667>.
- Suyuchen Wang, Jinlin Wang, Xinyu Wang, Shiqi Li, Xiangru Tang, Sirui Hong, Xiao-Wen Chang, Chenglin Wu, and Bang Liu. Improving context fidelity via native retrieval-augmented reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025a.
- Xinyu Wang, Linrui Ma, Jerry Huang, Peng Lu, Prasanna Parthasarathi, Xiao-Wen Chang, Boxing Chen, and Yufei Cui. Resona: Improving context copying in linear recurrence models with retrieval. In *Conference on Language Modeling (COLM)*, 2025b.
- Xinyu Wang, Sicheng Lyu, Yu Gu, Jerry Huang, Peng Lu, Yufei Cui, and Xiao-Wen Chang. Beyond hard writes and rigid preservation: Soft recursive least-squares for lifelong LLM editing. *arXiv preprint arXiv:2601.15686*, 2026.
- Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem- $\{\alpha\}$: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025c.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Jason Weston and Sainbayar Sukhbaatar. System 2 attention (is something you might need too). *arXiv preprint arXiv:2311.11829*, 2023.
- Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.
- 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. *CoRR*, abs/2412.15115, 2024a. doi: 10.48550/ARXIV.2412.15115. URL <https://doi.org/10.48550/arXiv.2412.15115>.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jian Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *CoRR*, abs/2505.09388,

- 2025a. doi: 10.48550/ARXIV.2505.09388. URL <https://doi.org/10.48550/arXiv.2505.09388>.
- An Yang, Bowen Yu, Chengyuan Li, et al. Qwen2.5-1m technical report, 2025b. URL <https://arxiv.org/abs/2501.15383>.
- Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024b.
- Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025c. URL <https://openreview.net/forum?id=r8H7xhYPwz>.
- 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 Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1259. URL <https://doi.org/10.18653/v1/d18-1259>.
- Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiyang 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, 2025. URL <https://arxiv.org/abs/2507.02259>.
- Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. Large language models are semi-parametric reinforcement learning agents. *Advances in Neural Information Processing Systems*, 36:78227–78239, 2023.
- Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiayi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture. *CoRR*, abs/2510.26692, 2025. doi: 10.48550/ARXIV.2510.26692. URL <https://doi.org/10.48550/arXiv.2510.26692>.
- Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025.

A Implementation

A.1 Prompts and Templates

We use two structured templates to implement the recurrent RETRIEVE–COMPRESS loop: a **Retriever Template** for decision making and query formation, and a **Memory Template** for faithful evidence compression.

Retriever Template (Figure 3) The retriever prompt conditions on the current question and the accumulated memory, and asks the model to (i) assess whether the memory already contains sufficient evidence to answer, and (ii) if not, produce a function-call specification for external retrieval. Concretely, the template outputs a discrete decision (STOP vs.

RETRIEVE); when retrieval is needed, it emits a search query and a top_k value. This design turns retrieval into an explicit, controllable action: the model is encouraged to issue broad queries when evidence is missing, refine queries when retrieval results are noisy or mismatched, and allocate top_k based on uncertainty (larger k when multiple candidate entities/facts exist; smaller k when the target is specific). By tying retrieval decisions to the evolving memory state, the agent can avoid redundant searches and terminate early once decisive evidence has been accumulated.

Memory Template (Figure 4) The memory prompt performs bounded, evidence-centric compression. At each step, it is given two sources: (1) the newly retrieved chunk (high-relevance but potentially noisy) and (2) a recurrent chunk from the running context (stable but may be redundant). The template instructs the model to extract only answer-relevant facts, normalize entities/aliases, and write a compact memory update that preserves verifiable evidence (names, dates, titles, and relations) while discarding stylistic or speculative content. Importantly, the template enforces *selective* compression across the two inputs: it prioritizes new complementary evidence from retrieval, but retains previously stored facts when they remain useful, preventing memory drift and uncontrolled growth.

A.2 Data Construction Details

Unified long-context synthesis pipeline. All synthesized long-context QA instances share the same supervision format: a question Q and an answer A , together with a set of *gold evidence documents* (or paragraphs) annotated by the source dataset. We convert each original instance into a *single long document* by mixing (i) the gold evidence documents, and (ii) a large pool of *distractor* documents sampled from the same corpus. Concretely, for each instance we build three text pools: the query (Q), the evidence set ($\mathcal{D}_{\text{gold}}$), and a distractor pool ($\mathcal{D}_{\text{dist}}$) drawn from the dataset’s training corpus.¹ We then create a candidate document list by shuffling documents with a fixed random seed, insert each gold document *exactly once* at the document level, and keep appending distractors until reaching a target token budget. This yields a *controlled* setting where (1) the answer is always supported by $\mathcal{D}_{\text{gold}}$, while (2) retrieval difficulty scales with the number of distractors and total context length.

A.2.1 Cold-start SFT data

Following the NIAH-style long-context QA construction in MemAgent, we synthesize cold-start SFT data from three QA sources: HotpotQA, SQuAD, and MuSiQue. Each source contributes 4,096 instances sampled from its training split. For each instance, we construct a long document at a fixed target length (32K tokens) by iteratively inserting distractor documents until the budget is met.² We use Qwen3-32B as the teacher with *thinking enabled* to generate protocol-consistent interaction traces under our PRETHINK–RETRIEVE–WRITE workflow: the teacher (i) plans and emits structured retrieve calls, (ii) updates a bounded agent memory by writing compressed evidence, and (iii) decides when to stop retrieving and answer. We then distill student backbones (Qwen3-1.7B, Qwen3-4B, and Qwen2.5-7B-Instruct) on these trajectories.

Question decompositions. MuSiQue provides an optional question decomposition (multi-hop sub-questions). We feed decompositions *only to the teacher* to elicit cleaner and more stable planning traces; students never observe decompositions, gold document IDs, or any teacher-side annotations during either training or inference. For HotpotQA and SQuAD, the teacher autonomously decides whether to decompose the question in its private reasoning and how to formulate retrieval queries.

¹We sample distractors from the same corpus to preserve domain/style match, making the task harder than using out-of-domain noise.

²In practice, we first *pre-scan* candidate distractor documents to determine how many whole documents can be inserted under a given token budget. We then construct the long document in a single pass by inserting all gold evidence documents once and appending the maximal number of distractors without exceeding the target length, truncating only at document boundaries.

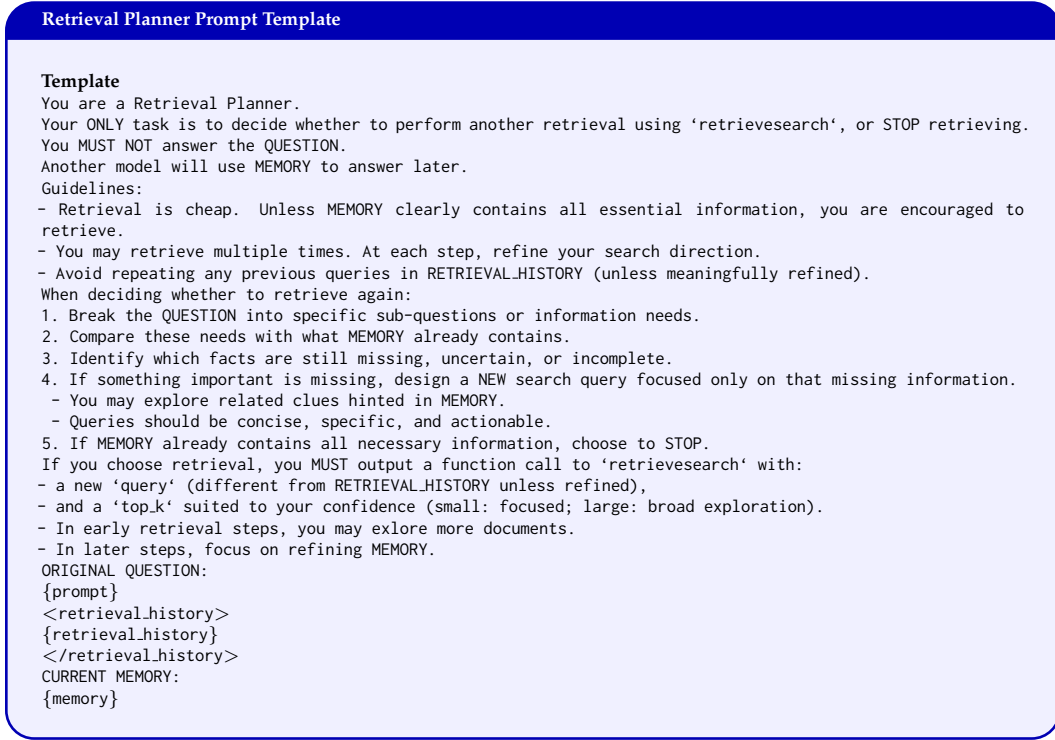


Figure 3: Prompt template for the Retrieval Planner, which decides whether to call retrievesearch again or stop, without answering the question.

Trajectory filtering. To ensure supervision quality, we retain only traces whose final answers are correct under the official evaluation protocol of the underlying dataset and discard all failed attempts. We additionally remove excessively long traces that would exceed the memory budget or truncate the agent memory/state; this ensures the student is trained on trajectories that are feasible at inference time under the same bounded-memory constraints.

After this filtering process, we decompose the successful trajectories into individual turns, resulting in a total of 29,717 single-turn dialogue instances. These instances constitute our final SFT dataset for training the student backbones.

A.2.2 RL training data.

For RL training, we utilize the same synthesis pipeline to extend the context length of HotpotQA instances to approximately 28K tokens. We retain the original question-answer pairs while scaling the retrieval difficulty through the insertion of distractors. During the reinforcement learning phase, the model is optimized using the Exact Match (EM) score between the generated response and the ground-truth answer as the primary reward signal. This setup ensures that the environment remains consistent with our SFT stage, allowing the RL process to focus specifically on refining the agent’s decision-making—such as retrieval timing and memory management—under long-context constraints.

A.2.3 Evaluation Benchmark

Synthesized long-context QA benchmarks (extreme scaling). To evaluate robustness under extreme context scaling, we create long-document variants following the NIAH-style construction for representative multi-hop QA tasks, including HotpotQA, 2WikiMulti-hopQA, and MuSiQue; we also include the synthetic SQuAD setting used in MemAgent for direct comparison. We use each dataset’s *test split* and sample 128 instances per task.

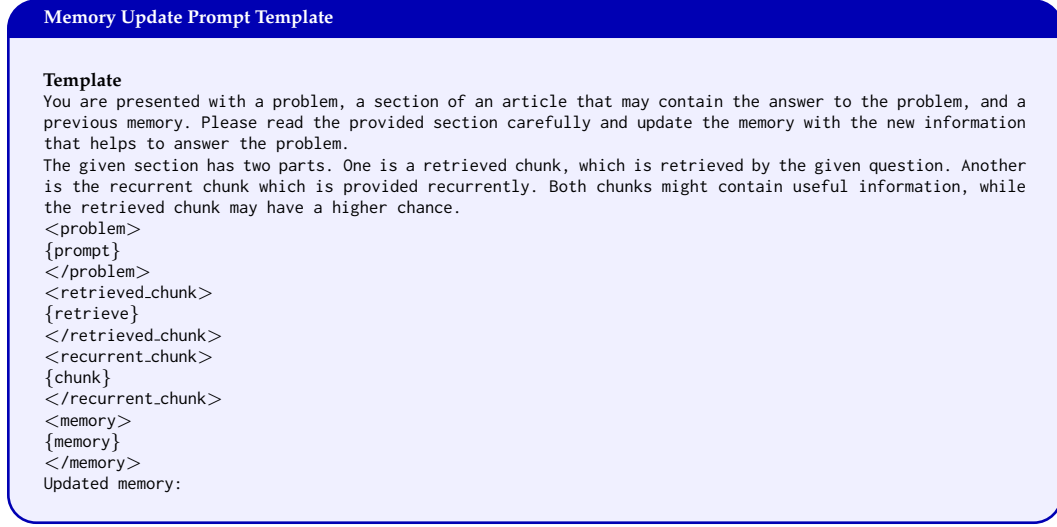


Figure 4: Prompt template for memory updating, integrating both retrieved and recurrent chunks to refine the memory state.

For each fixed question set, we generate multiple test variants at increasing target lengths (e.g., 32K/28K, 64K/56K, 128K/112K, up to 1M/896K tokens) by progressively inserting more distractors while keeping the gold evidence set unchanged. Gold evidence is inserted once per instance at the document level with a fixed seed, and distractors are sampled from the same corpus to preserve distributional match. This protocol ensures that differences across lengths reflect only the effect of *context scaling* (more distractors / longer inputs), not changes in questions or evidence.

Task-specific token budgets. The minimum target length differs slightly across tasks: HotpotQA uses 28K tokens to match the document-count-based construction inherited from the RL dataset, while other tasks use fixed token budgets (32K/64K/128K/.../1M) and insert as many whole documents as allowed under each budget.

LongBench QA benchmarks (natural distributions). To verify transfer beyond synthetic distractor insertions, we additionally evaluate on LongBench QA using its original documents and distributions. We report F1 on NarrativeQA, HotpotQA, 2WikiMultihopQA, Qasper, and MuSiQue following the official LongBench evaluation protocol.

A.3 Training setup

A.3.1 Model Configuration and Baselines

To ensure a rigorous evaluation, we standardize the recurrent interaction settings across both INFMEM and the baseline MEMAGENT (based on Qwen-1.5B/4B backbones).

Recurrent Processing Setup. Both models operate with a fixed **recurrent chunk size of 5,000 tokens**. To maintain consistency in the reasoning horizon, we align the **maximum generation length** (1.5k tokens) and the interaction iteration steps for both models. For INFMEM, we enable BM25-based retrieval with a cap of 4,000 retrieved tokens per step. Crucially, during the memory update phase of INFMEM, we explicitly filter out reasoning/thinking steps, retaining only the schema-consistent memory tokens to maximize information density.

Baseline Fairness. For the MEMAGENT reproduction, we disable the optional “thinking mode” (as discussed in §E.1) to adhere to its standard efficient setting. It is important to note that our comparison aligns the *output* constraints (generation length and steps) rather than

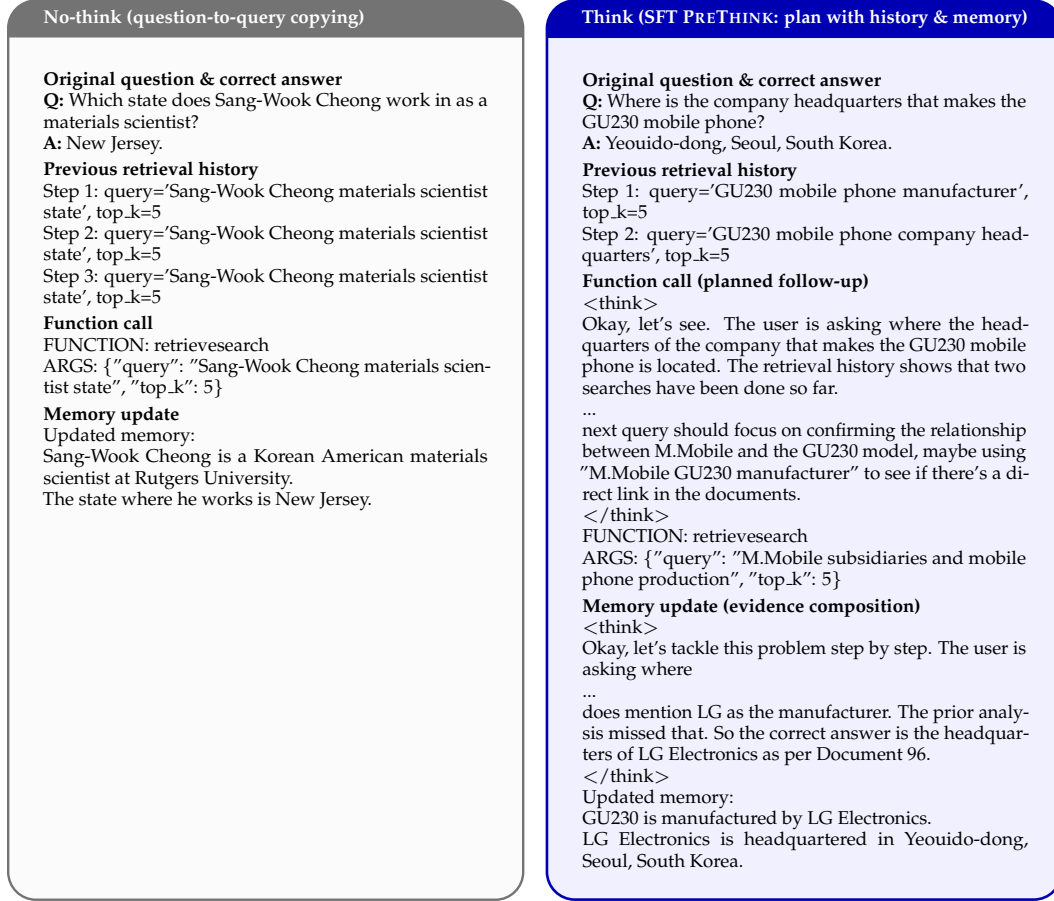


Figure 5: **Visualized retrieval trajectories: No-think vs Think.** Without PRETHINK, the model tends to copy the question into repetitive queries. After SFT, the planner conditions on retrieval history and memory, identifies missing links, and issues a targeted follow-up query to complete multi-hop evidence composition.

the input/memory budget. Since INFMEM processes additional retrieved context (up to 4k tokens) within the same iteration framework, it is required to compress a significantly larger volume of information into the memory state compared to MemAgent. This setup ensures we are not weakening the baseline; rather, we are testing INFMEM's ability to handle higher information loads under strictly bounded generation resources.

B Baseline Implementation Details

To ensure a fair and rigorous comparison against our proposed methods, we carefully reconstructed the experimental setups for the strong agentic baselines, RESP and Mem- α . Because these methods were originally designed for different scales or exhibited generation bottlenecks, we integrated a larger Qwen3-32B backbone to maximize their competitive performance. We also standardized the configurations for our standard long-context and retrieval baselines.

RESP: We strictly follow the implementation details provided in the original paper. To accurately replicate their iterative RAG logic, we utilized their exact prompt templates. We employed Qwen3-32B as the backbone model throughout the entire process, handling the iterative reasoning, planning, summarization, and the final generation.

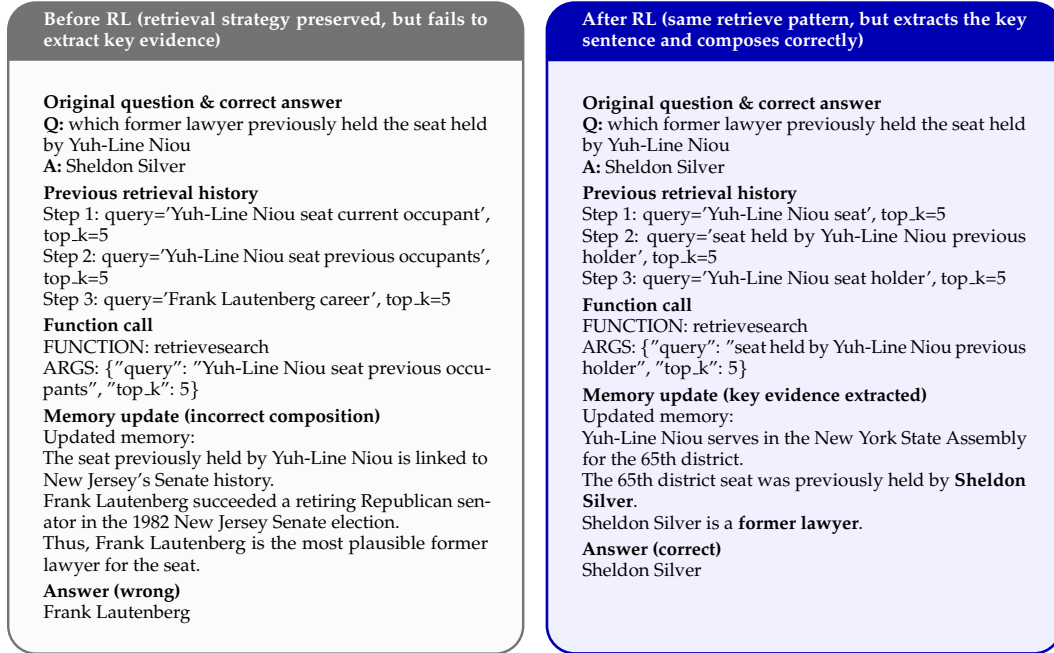


Figure 6: **RL effect on evidence extraction and memory writing.** Both runs use a similar retrieval pattern, but *before RL* the agent fails to identify the direct sentence linking Niou’s seat to the previous holder and instead hallucinates an unrelated political chain. *After RL*, the agent reliably extracts the decisive evidence (Niou → NY Assembly 65th district → Sheldon Silver) and writes a compact, answer-ready memory.

Mem- α : We integrated the publicly available Mem- α implementation directly into our evaluation framework. Specifically, we utilized their released 4B checkpoint to serve as the context manager. However, empirical observations indicated that the RL-tuned Mem- α checkpoint exhibited suboptimal generative question-answering capabilities. To prevent the baseline’s performance from being unfairly bottlenecked by poor generation rather than poor context management, we decoupled the architecture: we delegated the final QA generation phase to Qwen3-32B while retaining the 4B model strictly for context management.

YaRN: For evaluations utilizing YaRN, we apply the official RoPE scaling configurations established for the Qwen2.5 and Qwen3 models. Specifically, we set the rope_type to yarn, apply a scaling factor of 4.0, and maintain the original_max_position_embeddings at 32,768.

RAG: For the standard RAG baseline, our reported main results utilize a chunk size of 1,000 tokens and retrieve the top-6 chunks. To determine this optimal configuration, we conducted an ablation study on the synthesized HQA benchmark across context lengths of 28k, 56k, and 112k using Qwen3-4B. We evaluated multiple retrieval variants, including top-12 with 1,000-token chunks, top-6 with 2,000-token chunks, top-2 with 1,000-token chunks, and top-6 with 1,000-token chunks. Empirical results demonstrated that retrieving the top-6 chunks with a 1,000-token chunk size consistently yielded slightly better performance, and it was thus selected as the default configuration for all RAG evaluations.

C Case Study

C.1 sft enhance dieversity

Why SFT warmup is necessary. SFT is critical for making tool-use *reliable* in our agentic retrieval loop. In practice, base backbones do not consistently exhibit disciplined query

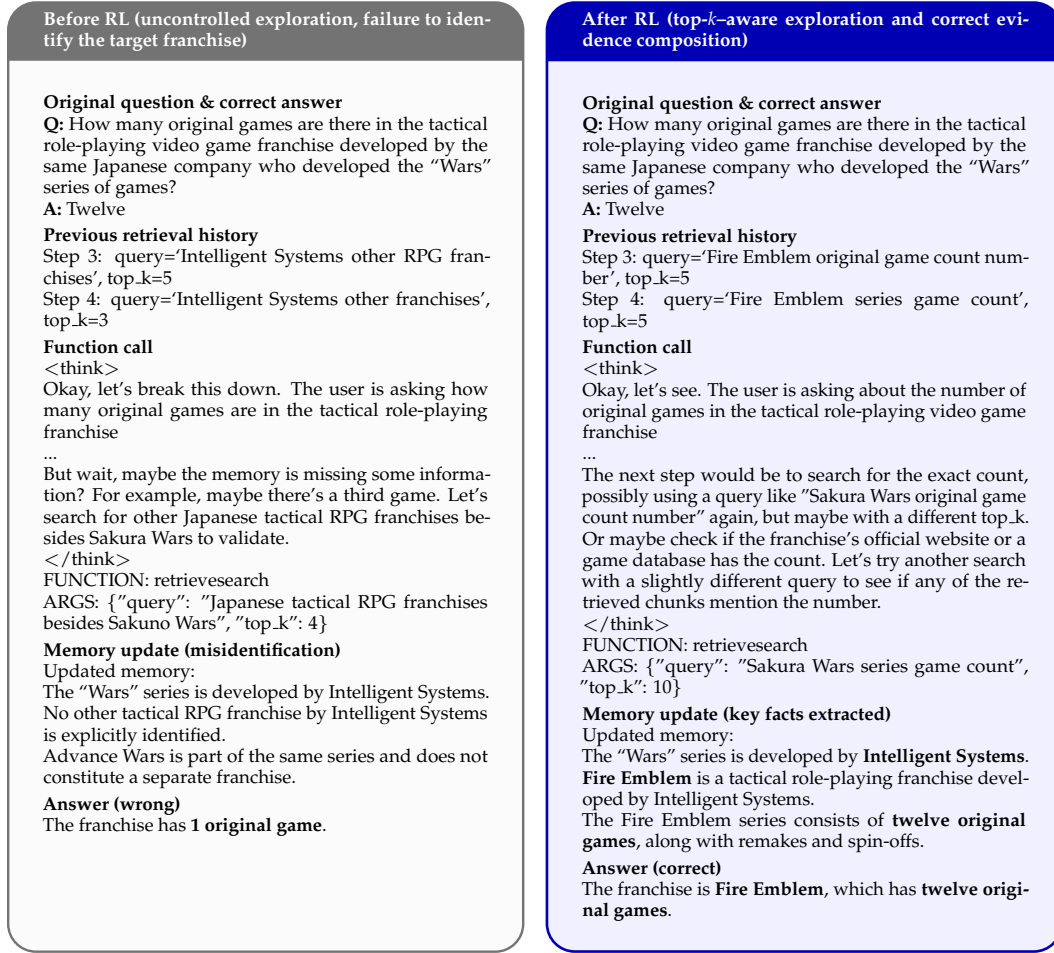


Figure 7: **Effect of RL on retrieval control and multi-hop reasoning.** Before RL, the agent fails to regulate exploration and prematurely concludes that no other tactical RPG franchise exists beyond the “Wars” series. After RL, the agent learns to adapt the retrieval scope via top-k control, successfully identifies *Fire Emblem* as the relevant franchise, and composes the correct numerical answer from explicit evidence.

planning: the smaller Qwen3-1.7B has limited intrinsic reasoning capacity, while the instruction-tuned Qwen2.5-7B still fails to reliably trigger deliberate multi-step planning under our RETRIEVE-COMPRESS protocol. Empirically, Table 5 and Table 6 shows consistent improvements from Train-free to SFT across both LB and RULER-QA, and even the strongest backbone (Qwen3-4B) benefits substantially, suggesting that supervised warmup improves not only downstream QA accuracy but also the quality of intermediate actions.

Qualitatively, Figure 5 visualizes retrieval trajectories on the same instance: without PRE-THINK, the model often degenerates into copying the question (or a lightly rewritten variant) as the search query, leading to repetitive, low-information retrievals. After SFT, the planner conditions on retrieval history and the current memory state, identifies missing links needed for multi-hop composition, and issues targeted follow-up queries, yielding more informative function calls and more dependable evidence aggregation.

Table 4: Additional details of InfMem inference protocol.

(a) InfMem inference algorithm

Algorithm 1: InfMem Inference Protocol

Input: question q ; streaming chunks $\{c_t\}_{t=1}^T$; global retrieval units $\{p_j\}_{j=1}^N$; budget M
Initialize: memory $m_0 \leftarrow \emptyset$
for $t = 1$ **to** T **do**
 // Step 1: Monitor & Plan (PreThink)
 $(a_t, u_t, k_t) \leftarrow \text{PRETHINK}(q, m_{t-1})$
 if $a_t = \text{STOP}$ **then**
 break // Early stopping triggered
 end if
 // Step 2: Seek (Retrieve)
 if $a_t = \text{RETRIEVE}$ **then**
 $r_t \leftarrow \text{RETRIEVE}(u_t, k_t; \{p_j\})$
 end if
 // Step 3: Update (Write with Joint Compression)
 $m_t \leftarrow \text{WRITE}(q, m_{t-1}, c_t, r_t; M)$
end for

 // Final Answer Generation
 $\hat{y} \leftarrow \text{ANSWER}(q, m_{\text{final}})$

(b) Design rationale of InfMem components

Component	Rationale
PRETHINK	Acts as a state-dependent controller to monitor sufficiency and plan query u_t based on memory m_{t-1}
RETRIEVE	Enables global, non-monotonic access to sparse evidence $\{p_j\}$ missed by linear scanning
WRITE	Performs evidence-aware <i>joint compression</i> , prioritizing bridging links from both c_t and r_t
EARLY STOP	Terminates inference once evidence is sufficient ($a_t = \text{STOP}$), reducing latency and redundancy

C.2 RL boost the performance

RL further strengthens InfMem beyond SFT by explicitly optimizing *long-horizon* tool-use under verifiable QA rewards, yielding gains along two complementary axes: **(i) memory compression / evidence writing**, and **(ii) planning / retrieval control**.

As shown in Figure 6, the pre-RL agent may follow a superficially similar retrieval pattern, yet fails at the decisive step of *extracting and committing* the key sentence into memory: rather than grounding “Niou’s seat” in the exact district that links to the previous holder, it writes an unrelated political chain and hallucinates an incorrect former lawyer. After RL, the agent consistently identifies the decisive evidence chain (Niou $\rightarrow$ NY Assembly 65th district $\rightarrow$ Sheldon Silver) and writes a compact, answer-ready memory, enabling a correct final answer.

Figure 7 highlights a complementary improvement in *planning*: RL teaches the agent to regulate exploration by adapting retrieval scope (e.g., via top- k control) instead of drifting or stopping prematurely. Before RL, the agent prematurely concludes that no tactical RPG franchise exists beyond the “Wars” series; after RL, it expands search when uncertain,

Table 5: **InfMem results on LongBench QA (LB) and RULER-QA.** We report per-task LB scores (NQA, HQA, 2Wiki, Qasper, MuSiQue), along with **avg.LB** and **avg.RULER-QA**.

Setting	Model	LB NQA	LB HQA	LB 2Wiki	LB Qasper	LB MuSiQue	avg.LB	avg.RULER-QA
Train-free								
	Qwen3-1.7B	20.25	48.73	54.05	33.91	28.40	37.07	37.71
	Qwen2.5-7B	19.76	52.95	48.78	31.09	31.69	36.85	47.96
	Qwen3-4B	23.27	60.96	69.66	35.14	44.19	46.64	50.25
SFT								
	Qwen3-1.7B	18.12	47.88	46.97	31.90	31.25	35.22	43.72
	Qwen2.5-7B	19.95	56.46	63.23	35.31	40.07	43.00	49.30
	Qwen3-4B	18.71	62.19	72.13	36.09	44.90	46.80	54.86
RL								
	Qwen3-1.7B	19.23	59.28	55.02	33.19	40.98	41.54	50.84
	Qwen2.5-7B	20.43	60.34	65.19	35.68	50.66	46.46	59.53
	Qwen3-4B	20.77	65.14	74.76	40.74	53.22	50.93	66.40

Table 6: **Performance gains across training stages across model scales.** Values in parentheses indicate absolute improvements (Δ) over the preceding step. Base Model is the starting point.

Model	Base Model		Yarn		Train-free		SFT		RL	
	avg.LB	avg.RULER	avg.LB	avg.RULER	avg.LB	avg.RULER	avg.LB	avg.RULER	avg.LB	avg.RULER
Qwen3-1.7B	25.64	2.64	32.61 (+6.97)	13.38 (+10.74)	37.06 (+4.45)	37.70 (+24.32)	35.22 (-1.84)	43.71 (+6.01)	41.54 (+6.32)	50.84 (+7.13)
Qwen2.5-7B	32.31	0.00	29.54 (-2.77)	21.41 (+21.41)	36.85 (+7.31)	47.95 (+26.54)	43.00 (+6.15)	49.30 (+1.35)	46.46 (+3.46)	59.53 (+10.23)
Qwen3-4B	38.39	4.30	39.46 (+1.07)	25.45 (+21.15)	46.64 (+7.18)	50.20 (+24.75)	46.80 (+0.16)	54.85 (+4.65)	50.92 (+4.12)	66.40 (+11.55)

discovers *Fire Emblem*, and composes the correct numerical answer from explicit evidence.

Taken together, these case studies suggest that RL does not merely increase tool usage; it trains the agent to *write the right information* into memory and to *plan the right next action*—balancing targeted exploration with timely stopping.

C.3 Early stop

Beyond accuracy, early stopping substantially improves inference efficiency. As shown in Fig. 8, once PRETHINK determines that the required evidence is already present in memory, the agent explicitly terminates the recurrent retrieve–write loop. This allows the model to exit inference as soon as it is confident in the answer, rather than continuing unnecessary iterations over the remaining context. As a result, inference time is no longer proportional to the document length or number of chunks (i.e., avoiding the typical $O(n)$ recurrent generation cost), and instead approaches constant-time behavior in practice when decisive evidence is found early.

D Ablation

D.1 Retrieval Chunk Size Selection

We study the effect of retrieval chunk size under a fixed retrieval budget in Table 7. Specifically, we constrain the total retrieved context to approximately 3k tokens and vary the chunk size and corresponding top- k : chunk=250, top- k = 12, chunk=500, top- k = 6, chunk=1000, top- k = 3, chunk=2000, top- k = 2, and chunk=3000, top- k = 1. Table 7 reports accuracy on long-context QA benchmarks (HQA 28k, SQuAD 32k, MuSiQue 32k, 2Wiki 32k) as well as LongBench QA.

Table 7: **Fixed-budget comparison across retrieval chunk sizes.** Under a constant retrieval budget, we vary the retrieved chunk size and report accuracy on long-context QA (HQA_28k, SQD_32k, MSQ_32k, 2WK_32k) and LongBench QA (Avg_LB with per-task scores). Overall, chunk size 500 achieves the best Avg_RULER and Avg_LB, suggesting a favorable balance between retrieval granularity and content diversity.

Setting	Avg_RULER	HQA_28k	SQD_32k	MSQ_32k	2WK_32k	Avg_LB	NQA	HQA	2Wiki	Qasper	MuSiQue
chunk_250_top12	53.96	55.77	61.27	38.40	60.41	45.72	21.09	60.63	68.62	34.79	43.48
chunk_500_top6	55.15	60.45	60.61	38.74	60.80	48.27	24.19	61.56	68.95	35.29	51.37
chunk_1000_top3	53.65	58.52	61.05	35.75	59.26	46.37	22.19	57.84	71.97	35.01	44.84
chunk_3000_top1	44.59	50.37	47.43	30.92	49.64	45.68	19.98	60.19	72.82	36.14	39.29
chunk_2000_top2	49.46	51.92	54.05	37.77	54.08	47.47	22.68	60.97	72.12	38.86	42.71
chunk_1000_top4	49.98	54.41	55.07	34.04	56.39	45.71	22.09	58.50	70.21	34.44	43.30

Table 8: Comparative Analysis of Memory Dynamics and Downstream Performance. The table reports Found rate, Preserved rate, and overall Performance across four datasets with varying context lengths.

		Avg	HQA			SQD			MSQ			2WK		
Model	Metric	-	28k	56k	112k	32k	64k	128k	32k	64k	128k	32k	64k	128k
MemAgent	Found	76.04	81.25	79.69	78.12	89.84	92.97	91.41	56.25	52.34	57.81	79.69	75.78	77.34
	Think-RL Preserved	65.43	72.66	67.97	61.72	78.12	85.16	78.12	47.66	44.53	42.97	73.44	67.19	65.62
MemAgent NoThink-RL	Found	72.46	81.25	76.56	74.22	90.62	95.31	91.41	63.28	49.22	50.78	63.28	64.06	69.53
	Preserved	69.53	78.12	74.22	72.66	89.84	95.31	91.41	60.16	44.53	46.88	57.03	60.94	63.28
	Performance	56.94	71.18	66.21	62.42	69.49	69.84	72.96	41.79	41.55	36.62	56.43	48.55	46.18
InfMem	Found	74.61	75.78	71.88	75.00	78.91	78.91	78.91	60.16	64.06	61.72	85.16	82.81	82.03
	Preserved	69.77	72.66	69.75	72.16	71.09	72.66	72.66	57.03	59.38	59.38	76.56	76.56	77.34
	Performance	64.85	70.03	69.34	71.36	65.44	62.23	68.11	52.77	56.59	55.59	70.55	67.22	68.91

Overall, a chunk size of 500 tokens achieves the best or near-best performance across most tasks. Very small chunks (e.g., 250 tokens) provide fine-grained retrieval but can fragment semantically coherent evidence, increasing the burden on memory composition and cross-chunk reasoning. Conversely, large chunks (e.g., 2000–3000 tokens) preserve local coherence but reduce content diversity under a fixed budget, increasing the risk that irrelevant context dilutes the decisive evidence. The intermediate setting (chunk=500, top- k = 6) strikes a favorable balance between retrieval granularity and evidence coverage, enabling InfMem to capture complementary facts while maintaining sufficient local context for reliable extraction and memory writing.

We additionally test a larger retrieval budget of approximately 4k tokens by including chunk=1000, top- k = 4 and chunk=2000, top- k = 2. The same trend persists: the 500-token regime remains a robust sweet spot, suggesting that the optimal chunk size is primarily governed by the trade-off between granularity and diversity rather than the exact budget.

D.2 Early Stop & Latency Analysis

To investigate the real-world efficiency and performance of the InfMem framework, we evaluate our stopping policies on a subset of 128 test samples across the benchmarks. We deploy the models using vLLM to measure the actual serving latency alongside accuracy. Table 10 summarizes the time and quality metrics corresponding to the visualization in Figure 2. The results show that while the 1-stop variant offers the lowest latency, it suffers from performance degradation due to the premature truncation of evidence chains. In contrast, our default 3-stop variant occupies the Pareto frontier. It balances negligible computational overhead with significantly higher answer accuracy, confirming that a slightly conservative stopping policy preserves critical evidence without sacrificing practical serving efficiency.

To further characterize this operational efficiency, Table 16 provides a detailed breakdown of the step distribution. Our method significantly reduces the total number of iteration steps through a highly adaptive and intelligent early stopping mechanism. InfMem autonomously modulates its search depth based on task complexity. For single-hop tasks like SQuAD,

Table 10: **Effect of early-stopping strategies on performance and wall-clock time.** We compare MemAgent (baseline) with two early-stop variants (1-stop and 3-stop) across three backbones. Columns report Avg, HotpotQA (HQA), SQuAD, MuSiQue, and 2WikiMultihopQA (2Wiki), with *Perf.* shown on the first row and *Time* on the second row for each model.

Model	Metric	MemAgent					InfMem 3-stop					InfMem 1-stop				
		Avg	HQA	SQuAD	MuSiQue	2Wiki	Avg	HQA	SQuAD	MuSiQue	2Wiki	Avg	HQA	SQuAD	MuSiQue	2Wiki
7B	Perf.	52.13	58.39	68.63	38.34	43.18	63.00	57.51	67.71	58.57	68.20	59.86	54.01	70.10	51.56	63.76
	Time	51:34	43:48	50:09	54:37	57:44	21:35	28:10	19:00	18:19	20:49	15:46	21:16	13:27	14:13	14:08
1.7B	Perf.	36.59	42.50	47.29	24.05	32.52	49.35	51.31	59.56	38.18	48.34	48.39	54.52	53.39	36.19	49.45
	Time	41:51	37:45	41:06	43:16	45:18	20:50	16:33	18:20	28:41	19:46	12:41	11:03	10:52	16:20	12:28
4B	Perf.	50.13	51.70	77.74	35.91	35.18	65.80	66.13	73.81	56.86	66.39	61.80	62.91	66.19	50.45	67.65
	Time	60:45	51:31	64:09	59:37	67:44	23:59	27:33	18:40	19:00	30:42	11:49	15:19	9:29	12:41	9:45

the framework rapidly converges on direct evidence, resulting in the shortest average step count. Conversely, for complex multi-hop tasks like MuSiQue, it deliberately traverses the document pool to resolve interdependent sub-queries. The early stop decision acts as a smart exit strategy: it terminates in very few steps when key evidence is quickly identified, yet retains the capacity to recurse deeply when initial searches are suboptimal, thereby optimizing execution speed without compromising the integrity of the evidence chain.

Table 11 demonstrates the average total time proportion of different components per sample. Although our controller introduces new decision-making & searching steps, building the BM25 index and the actual retrieval process consume only a marginal fraction of the total time. The substantial early stopping mechanism drastically cuts down the overall average latency and the time spent updating memory. Furthermore, as analyzed in subsequent sections, our framework generates higher-quality memory with less repetition; consequently, the time required to generate the final answer is also correspondingly reduced. Interestingly, when monitoring the theoretical average latency per sample, the speedup ratio over baselines is even more pronounced than the actual serving speedup. This discrepancy arises from the highly efficient continuous batching and queueing mechanisms of vLLM, which naturally optimize baselines like MemAgent that process a fixed number of chunks, whereas InfMem’s actual batch throughput is occasionally dragged down by a few outlier samples that require maximum exploration steps without early stopping.

Table 11: Average total time (s) across different context lengths. Speedup multipliers relative to memagent are shown as subscripts (teal for speedup > 1, red for speedup < 1). Components are abbreviated: Tot. = Total, B = bm25_build, J/P = judge_or_plan, R = retrieval, MU = memory_update, FA = final_answer.

Length	memagent				infmem 3 stop						infmem 1 stop					
	Tot.	MU	FA		Tot.	B	J/P	R	MU	FA	Tot.	B	J/P	R	MU	FA
28k	32.79	32.52	0.27		45.21 _{<0.73}	0.019	23.70	0.002	21.16 _{×1.54}	0.33 _{×0.81}	21.75 _{×1.51}	0.019	12.96	0.002	8.32 _{×3.91}	0.45 _{×0.60}
56k	71.86	71.52	0.35		51.77 _{<1.39}	0.037	28.62	0.003	22.81 _{×3.14}	0.30 _{×1.14}	36.60 _{×1.96}	0.037	22.48	0.003	13.83 _{×5.17}	0.25 _{×1.38}
112k	147.81	147.31	0.51		65.08 _{<2.27}	0.076	39.64	0.006	25.16 _{×5.86}	0.21 _{×2.44}	44.66 _{×3.31}	0.076	30.47	0.004	13.87 _{×10.62}	0.24 _{×2.16}
224k	331.35	330.82	0.52		93.48 _{<3.54}	0.156	61.69	0.010	31.39 _{×10.54}	0.23 _{×2.24}	72.95 _{×4.54}	0.156	53.24	0.009	19.27 _{×17.16}	0.27 _{×1.91}
448k	691.64	691.13	0.51		148.57 _{×4.66}	0.321	109.92	0.025	38.08 _{×18.15}	0.22 _{×2.28}	123.60 _{×5.60}	0.321	99.31	0.022	23.69 _{×29.17}	0.25 _{×2.05}
896k	1499.21	1498.60	0.61		311.08 _{×4.82}	0.676	225.41	0.109	84.45 _{×17.74}	0.43 _{×1.42}	229.50 _{×6.53}	0.676	190.34	0.055	38.08 _{×39.36}	0.35 _{×1.73}

E Ablation and Analysis of Thinking Dynamics

E.1 Rationale for Defaulting to No-Thinking in Baseline

When reproducing MemAgent-RL on the Qwen3-series, we adopt the *no-thinking* setting as the default configuration. This choice is primarily driven by **alignment with the original pipeline**, as the official MemAgent-RL setup (based on Qwen2.5-Instruct) does not inherently support thinking mode. However, beyond consistency, we empirically validate that enabling thinking in the baseline architecture is often counterproductive.

The baseline MemAgent relies on a lightweight controller for naive compression, deciding strictly whether to write or skip the current chunk. In this regime, enabling thinking mode

triggers an “over-deliberation” behavior. As illustrated in the case study (Figure 10), the agent expends substantial reasoning on loosely related chunks, which blurs the write/skip boundary.

Quantitative Evidence of Instability. This destabilization is quantitatively captured in Table 8. While enabling thinking in the baseline (*MemAgent Think-RL*) significantly improves the model’s ability to discover answers (Found: 76.04% vs. 72.46%), it introduces severe volatility. The active reasoning process makes the memory vulnerable to recurrent noise, causing the *Preserved* rate to drop sharply from 69.53% to 65.43%. This confirms that in the baseline architecture, the benefits of enhanced extraction are negated by the instability of memory updates, justifying our choice of the *no-thinking* configuration for controlled comparisons.

E.2 Decoupling Reasoning from Instability: The InfMem Advantage

Crucially, the analysis above raises a fundamental question: *Is reasoning inherently detrimental to memory retention in recurrent systems?* Our results with **InfMem** suggest the answer is no. The instability observed in the baseline stems not from the thinking process itself, but from the **naive compression** mechanism that fails to filter the generated reasoning paths.

Stability via Dynamic Chunking. As shown in Table 8, INFMEM effectively mitigates the extraction-retention trade-off. Despite leveraging reasoning to enhance information processing, INFMEM maintains a robust retention profile. Its average Preserved rate (69.77%) is not only significantly higher than the unstable *MemAgent Think-RL* but is fully comparable to the conservative *MemAgent NoThink-RL* baseline (69.53%). This indicates that INFMEM’s Dynamic Chunking successfully concentrates deliberation on salient regions, allowing the model to benefit from deep reasoning without succumbing to the forgetting issues typical of recurrent updates.

Memory Purity and Downstream Performance. The advantages of INFMEM extend beyond mere retention statistics to the **quality** of the preserved information (Fig 11). A key observation from Table 8 is the performance discrepancy: while INFMEM and *MemAgent NoThink-RL* preserve a similar number of answers ($\sim 69\%$), INFMEM achieves substantially higher downstream performance (64.85% vs. 56.94%).

F Extended Results

F.1 Generality Across Model Families

All backbones in the main results belong to the Qwen family. To test whether the control loop is tied to that family, we run the *identical* InfMem inference protocol on Gemma-4-E4B-IT, an instruction-tuned model of comparable effective size from a different training pipeline, *without any SFT or RL on that backbone*, and compare it against MemAgent on the same model. The evaluation setup follows the main comparison (HotpotQA at 112k–896k and MuSiQue at 128k–1M, 128 samples per length).

InfMem improves over MemAgent on all eight (task $\times$ length) configurations, by +9.1 to +24.1 F1 (Table 12), and reproduces the two qualitative patterns of the Qwen results: the gain is larger on multi-hop than on single-hop questions, and it grows with context length. Most directly relevant to the question of attribution, the average marginal lift on multi-hop MuSiQue is essentially the same on the two families (+20.3 on Gemma vs. +20.6 on Qwen3-4B), indicating that the benefit of the control loop is a property of the protocol rather than an artifact of Qwen-specific training. On single-hop HotpotQA the lift is somewhat larger on Gemma (+13.9 vs. +10.1), as the RL-trained Qwen baseline is already stronger on this task and leaves less headroom. The efficiency profile transfers as well, with up to $5.0\times$ lower per-sample latency at 896k (Table 13). We note that InfMem on the zero-shot Gemma backbone still trails InfMem on the RL-trained Qwen3-4B by roughly 3–5 F1 in absolute terms.

Table 12: **Zero-shot transfer of the InfMem control loop to a non-Qwen backbone.** We run the identical InfMem inference protocol (PRETHINK–RETRIEVE–WRITE, 3-stop, BM25) on Gemma-4-E4B-IT *without any SFT or RL on that backbone*, and compare against MemAgent on the same backbone. The last row reports the corresponding framework lift on the RL-trained Qwen3-4B, computed from Table 1, for reference. On multi-hop MuSiQue the marginal lift is nearly identical across the two families (+20.3 vs. +20.6), indicating that the gain is attributable to the control loop rather than to Qwen-specific training.

Backbone	Method	HotpotQA					MuSiQue				
		112k	224k	448k	896k	Avg	128k	256k	512k	1M	Avg
Gemma-4-E4B-IT (zero-shot)	MemAgent	57.81	51.84	50.96	45.47	51.52	19.60	26.80	19.90	23.22	22.38
	InfMem	66.91	64.41	67.08	63.20	65.40	39.45	41.16	43.99	45.98	42.65
	Δ	+9.10	+12.57	+16.12	+17.73	+13.88	+19.85	+14.36	+24.09	+22.76	+20.27
Qwen3-4B (RL)	Δ (reference)	+8.82	+8.30	+8.91	+14.43	+10.12	+19.00	+18.35	+23.95	+20.95	+20.56

Table 13: **Inference efficiency of the zero-shot transfer to Gemma-4-E4B-IT.** Left: per-sample latency (s) on HotpotQA. Right: total wall-clock (mm:ss, 128 samples, n_proc=64) on MuSiQue, together with InfMem’s *found-in-memory* rate. The efficiency profile matches the one observed on Qwen3-4B: the advantage grows with context length.

Ctx	HotpotQA (latency/sample, s)			MuSiQue (wall / found)		
	MemAgent	InfMem	Speedup	MemAgent	InfMem	Found
112k / 128k	109	63	1.7×	5:31	6:17	66.41
224k / 256k	225	91	2.5×	10:31	10:21	64.84
448k / 512k	593	134	4.4×	22:20	18:55	65.62
896k / 1M	1286	257	5.0×	54:22	43:13	64.06

Model-family dispatch. Supporting a non-Qwen backbone requires no change to the control loop, only to prompt rendering and response parsing. We add a tokenizer-based dispatch layer that detects the backbone family and selects the corresponding format: Qwen models use the native function-call markers, whereas Gemma uses the standard chat template with tool declarations and the model’s tool-call and thinking delimiters. Because those delimiters are special tokens, decoding for the Gemma path must preserve special tokens; the Qwen path is unchanged.

F.2 Additional Retrieval-based Baselines

Table 14 compares InfMem against two further families of retrieval-based systems under a matched in-document corpus.

Search-R1. Search-R1 is an RL-trained *retrieval-decision* policy, and is therefore the closest learned-control baseline to InfMem. We use the officially released qwen2.5-7b-em-ppo checkpoint and keep its trained <think>/<search>/<information>/<answer> loop intact, replacing only its open-domain Wikipedia/dense retriever with the same in-document BM25 index used by InfMem and MemAgent, so that the comparison isolates the quality of the search policy rather than the retrieval corpus. Search-R1 never places the long document in its context: it emits a query and receives the top-3 in-document chunks as <information>, so its effective context stays small and its scores are nearly flat across lengths. At the same backbone scale, it trails InfMem by 5.6 F1 on HotpotQA and 8.5 F1 on MuSiQue (averaged over the four lengths), and it is also below our 4B InfMem (53.9/46.4 vs. 68.1/58.4). It does clearly beat the no-control MemAgent baseline on multi-hop questions (46.4 vs. 39.2), confirming that a learned retrieval decision helps; the InfMem loop helps more.

Self-RAG. We evaluate Self-RAG in two forms. The first, denoted *prompted Self-RAG-style RAG*, is a train-free pipeline on Qwen3.6-35B-A3B (thinking mode) that uses the same in-

Table 14: **Additional retrieval-based baselines under a matched in-document corpus.** *Search-R1* keeps its officially released RL-trained `<think>/<search>/<information>/<answer>` policy (qwen2.5-7b-em-ppo) and only swaps its open-domain Wikipedia/dense retriever for the same in-document BM25 used by InfMem and MemAgent, so that the comparison isolates the search policy rather than the retrieval corpus. *Self-RAG-style (prompted)* is a train-free Self-RAG-style reflection pipeline on a larger MoE backbone. Bold denotes the best score per column within the 7B block.

Backbone	Method	HotpotQA					MuSiQue				
		112k	224k	448k	896k	Avg	128k	256k	512k	1M	Avg
Qwen2.5-7B	MemAgent	61.55	59.95	57.09	58.39	59.2	37.79	44.91	35.77	38.40	39.2
	Search-R1 (RL) + in-doc BM25	52.61	54.95	53.87	54.01	53.9	46.57	47.38	46.63	45.20	46.4
	InfMem	57.75	60.55	62.34	57.51	59.5	48.13	57.49	55.26	58.57	54.9
Qwen3.6-35B-A3B	Self-RAG-style (prompted, train-free)	46.48	43.05	41.26	42.13	43.2	13.52	11.25	10.42	8.22	10.9
Qwen3-4B	InfMem (full, for reference)	71.24	67.42	67.75	66.13	68.1	55.62	61.39	59.59	56.86	58.4

document BM25 retriever as our other baselines, prompts the model to emit Self-RAG-style reflection signals (retrieval, relevance, support and utility), and ranks candidate answers with the weighted arg-max rule of the original paper. Despite a much larger backbone, it stays 24–27 F1 below our 4B InfMem at every length on HotpotQA, and the gap widens sharply on multi-hop MuSiQue (13.5/11.3/10.4/8.2 F1 from 128k to 1M).

That collapse is localized to retrieval rather than to generation. The fraction of questions whose retrieved passages actually contain the gold answer drops from 46.1% on HotpotQA to 14.8% on MuSiQue, whereas the fraction whose reasoning trace mentions the answer barely moves (58.6% to 50.8%); consequently the model abstains outright (“not provided in the given documents”) on 63–66% of MuSiQue questions. Across both tasks the F1 of this baseline tracks its retrieval hit rate almost exactly (46.5 F1 at 46.1% hit rate on HotpotQA, 13.5 F1 at 14.8% on MuSiQue), i.e. a single question-shaped BM25 query essentially decides the outcome. This is precisely the failure mode that query reformulation prevents: Search-R1 is also retrieval-only and likewise never sees the full document, but because it re-issues queries across turns it loses only 11% relative when moving from HotpotQA to MuSiQue, against 71% for this Self-RAG-style pipeline.

The second is a Self-RAG generator we trained ourselves on Qwen3-4B-Instruct, following the official recipe (Alpaca-format prompts, the 15 reflection tokens with mean-initialized embeddings, paragraph-content loss masking, and the $\approx 150K$ official Self-RAG training instances), in order to rule out the possibility that the official checkpoints underperform only because they are built on older Llama-2 backbones. On Self-RAG’s in-domain single-hop setting, PopQA, evaluated with the official short-form script, our checkpoint reaches 0.60 accuracy and exceeds the official `selfrag_llama2.7b` checkpoint (0.54), confirming that it is correctly trained and competitive in the domain Self-RAG targets.

On our long-context benchmark, however, all Self-RAG variants remain far below InfMem (Table 15); we report sub-EM here because Self-RAG variants often emit reflection-token-style outputs rather than the short answers expected by our evaluator. We evaluated the official checkpoints under both inference algorithms of the original paper, short-form (single-step per-passage reranking) and long-form (multi-step beam search), and verified that they emit valid reflection tokens, so the gap is not a formatting artifact. The explanation is a task–paradigm mismatch rather than model capacity or checkpoint staleness: Self-RAG’s retrieve-then-critique paradigm assumes that a single retrieved passage is largely sufficient to answer, whereas our setting requires synthesis across passages. The short-form algorithm only ranks answers produced from individual passages, and the long-form algorithm performs no query reformulation, so it typically fails to retrieve second-hop evidence and effectively collapses to single-passage answering.

Table 15: **Self-RAG variants on synthesized long-context HotpotQA (sub-EM, %).** We report sub-EM because Self-RAG variants frequently emit reflection-token-style outputs and do not consistently follow the short-answer instruction used by our evaluator. Official checkpoints are evaluated under both inference algorithms of the original paper (short-form per-passage reranking and long-form beam search); we verified that they emit valid reflection tokens, so the low scores are not a formatting artifact.

Self-RAG variant	Backbone	800	6400
Official selfrag.llama2.7b	Llama-2-7B (trained)	13.3	17.2
Official selfrag.llama2.13b	Llama-2-13B (trained)	8.6	14.1
Prompted Self-RAG-style RAG	Qwen3.6-35B-A3B (train-free)	43.8	43.0
Trained Self-RAG (ours)	Qwen3-4B-Instruct (trained)	17.2	19.5

E.3 Token, Latency and Storage Accounting

Decode tokens and latency across lengths. Table 17 reports per-sample decode tokens together with per-sample latency for InfMem and MemAgent across six context lengths. Below 56k, the fixed PRETHINK overhead makes InfMem slightly more expensive in tokens, while its latency is already on par. From 112k onward InfMem dominates on both axes, saving 37–75% of the tokens and 56–79% of the wall-clock time.

As shown in Table 16, while the total number of chunks averages 202.8, InfMem 1-stop processes only 20.5 chunks on average (approximately 10% of the full context), and InfMem 3-stop requires just 32.6 steps (16%). This demonstrates that our adaptive stopping policy effectively identifies when sufficient information has been gathered, avoiding unnecessary computation over the remaining context.

Latency is saved faster than tokens (at 896k, $0.21\times$ the wall-clock but $0.28\times$ the tokens): MemAgent’s cost grows with the number of chunks it must stream, whereas InfMem keeps a constant-size working memory and issues only a few retrieval rounds, so its wall-clock grows about $10\times$ from 28k to 896k against about $46\times$ for MemAgent.

Retrieval and memory budget. Building and querying the index is a negligible fraction of the total cost (Table 11): InfMem issues about four retrieval calls per sample on average, and the BM25 build and lookup together account for well under 1% of per-sample latency even at 896k. The working memory is bounded by construction, at a 5k-token recurrent chunk plus a 1k-token memory slot per step, and is therefore constant in the input length.

Index storage. Because retrieval is performed over a per-document index, we also report its storage footprint (Table 18). With a production-grade Pyserini/Lucene backend, the minimal index needed for retrieval is $0.25\text{--}0.35\times$ the size of the raw text, and the ratio *decreases* with document length, so the per-token cost is bounded: the index of a 914k-token document is only 885 KB. Storing positions, doc-vectors and raw fields raises this to about $1.8\times$ raw, still below the $2.7\text{--}2.9\times$ of the uncompressed rank_bm25 reference implementation used in our main experiments.

Table 16: Step-wise statistics of the InfMem variants across various benchmarks. The gap between average and maximum steps highlights the efficiency of the adaptive stopping policy.

Metric	Avg. (All)	HQA	SQuAD	MuSiQue	2Wiki
Total Chunks	202.8	181.1	209.8	210.0	210.1
3-Stop Avg. Steps	32.6	27.9	20.1	51.1	31.0
Range (Min–Max)	4–205	4–183	4–210	4–212	4–213
1-Stop Avg. Steps	20.5	15.2	9.6	35.1	22.2
Range (Min–Max)	2–204	24–183	2–210	2–211	2–212

Table 17: **Per-sample decode tokens and latency of InfMem (3-stop) vs. MemAgent across context lengths (Qwen3-4B, 128 samples per length, 4×H100).** Below 56k, the fixed PRETHINK overhead makes InfMem more expensive in tokens while latency is already on par; from 112k onwards InfMem dominates on both axes, and latency is saved faster than tokens, since MemAgent’s per-step cost grows with the number of chunks whereas InfMem keeps a constant working memory plus a few retrieval rounds.

Len	Ctx	MemAgent		InfMem		Ratio	
		tok/Q	s/Q	tok/Q	s/Q	tok	s
200	~28k	2,192	32.8	4,422	45.3	2.02×	1.38×
400	~56k	5,378	71.9	5,308	51.9	0.99×	0.72×
800	~112k	10,697	148.0	6,728	65.3	0.63×	0.44×
1600	~224k	25,057	331.7	9,173	93.9	0.37×	0.28×
3200	~448k	52,861	692.5	13,154	149.5	0.25×	0.22×
6400	~896k	117,418	1501.0	33,085	312.9	0.28×	0.21×

Table 18: **Storage footprint of the in-document BM25 index** (chunk size 500, overlap 100; averaged over 16 samples per length). *Lucene min* is a production-grade Pyserini/Lucene index with only the fields needed for retrieval; *Lucene full* additionally stores positions, doc-vectors and raw fields; *rank_bm25* is the uncompressed pure-Python reference implementation used in our main experiments. The index-to-raw ratio *decreases* with document length, so the per-token storage cost is bounded: a 914k-token document needs only 885 KB.

Len	#Chunks	Ctx tok	Raw	Lucene min		Lucene full		rank_bm25	
				size	/raw	size	/raw	size	/raw
800	281	112k	442.6 KB	156 KB	0.35×	857 KB	1.94×	1.3 MB	2.93×
1600	568	227k	894 KB	276 KB	0.31×	1.6 MB	1.88×	2.5 MB	2.85×
3200	1140	456k	1.7 MB	489 KB	0.27×	3.2 MB	1.84×	4.8 MB	2.78×
6400	2284	914k	3.5 MB	885 KB	0.25×	6.3 MB	1.80×	9.5 MB	2.73×

Early Stop (with PRETHINK: decide retrieval vs. stop)

Original question & correct answer

Q: Which American composer provided music for the musical 'Something More!'?

A: Sammy Fain

Iteration 1

<think>

Okay, let's tackle this step by step. The user is asking which American composer provided music for the musical 'Something More!'. First, I need to break down the question into sub-questions.

...

if the results are clear, the answer can be extracted. If not, further refinements might be needed. But given the current state, the first search is essential to proceed.

</think>

FUNCTION: retrievsearch

ARGS: {"query": "American composer for Something More!", "top.k": 5}

Memory update

Updated memory: - The retrieved and recurrent chunks do not provide information about an American composer for "Something More!". - The band Syren is mentioned, but their album title is "Something Has To Separate," not "Something

Iteration 2

<think>

Okay, let's see. The user is asking which American composer provided music for the musical 'Something More!'.

...

but since the user's question is about the musical "Something More!", the next step is to search for the band Syren and their album title to see if the composer is part of that.

...

</think>

FUNCTION: retrievsearch

ARGS: {"query": "Syren Something Has To Separate composer", "top.k": 5}

Memory update

The provided retrieved and recurrent chunks do not explicitly mention an American composer associated with the musical "Something More!". The closest reference is to the band Syren and their album "Something Has To Separate", but this title is distinct from "Something More!". No composer is directly linked to "Something More!" in the given documents.

Updated memory: - The retrieved and recurrent chunks do not provide information about an American composer for "Something More!". - The band Syren is mentioned, but their album title is "Something Has To Separate," not "Something More!". - No composer is explicitly linked to "Something More!" in the provided documents.

The answer may not be present in the current retrieved or recurrent chunks, but further context or clarification on the title discrepancy (e.g., a typo or alternate title) might be needed to identify the correct composer.

Iteration 3

<think>

Okay, let's see. The user is asking which American composer provided music for the musical 'Something More!'. The retrieval history shows that they've already done four steps of searching.

...

The current memory doesn't have that info, so the next step is to retrieve again with a query that includes "Something Has To Separate" and "composer" to get the necessary information.

</think>

FUNCTION: retrievsearch

ARGS: {"query": "Something Has To Separate composer", "top.k": 5}

Memory update

The retrieved chunk (Document 109) explicitly mentions that the musical "Something More!" has music composed by **Sammy Fain** (with lyrics by Marilyn Bergman and Alan Bergman) and additional contributions from Composer Robert Prince. This directly answers the problem.

Updated memory: - The musical "Something More!" is composed by **Sammy Fain**, as stated in Document 109. - The band Syren's album "Something Has To Separate" is a distinct work and unrelated to "Something More!". - Composer Robert Prince also contributed to the musical's dance numbers.

The answer is **Sammy Fain**. **PRETHINK: early-stop decision**

<think>

Okay, let's see. The user is asking which American composer provided music for the musical 'Something More!'.

...

which is a different work. So the answer is already in the memory. No need for another retrieval. The answer is Sammy Fain.

</think>

STOP

Answer

Sammy Fain

Figure 8: **Case study: early stopping enabled by PRETHINK.** The agent uses explicit planning to decide whether to retrieve or stop. It first issues broad queries that fail due to a title mismatch (confusing the musical *Something More!* with an unrelated album), then refines the query and retrieves decisive evidence that the musical's music is by **Sammy Fain**. Once the required fact is present in memory, PRETHINK triggers *STOP* to avoid redundant searches and unnecessary memory overwrites.

	MemAgent (No-think)	MemAgent (Think)	InfMem
avg	<u>59.72</u>	58.99	66.00
HQA_28k	<u>71.18</u>	59.20	71.44
SQD_32k	69.49	61.70	<u>65.31</u>
MSQ_32k	41.79	<u>46.27</u>	56.58
2WK_32k	56.43	<u>68.78</u>	70.66
avg_LB	<u>47.11</u>	46.46	50.93
LB NQA	<u>20.74</u>	20.43	20.77
LB HQA	<u>63.80</u>	60.34	65.14
LB 2Wiki	<u>67.83</u>	65.19	74.76
LB Qasper	<u>41.02</u>	35.68	40.74
LB Musique	42.14	<u>50.66</u>	53.22

Table 9: **Thinking-mode ablation for reproducing MemAgent-RL on Qwen3-4B.** We compare Qwen3-4B with *thinking mode* enabled vs. disabled when reproducing the MemAgent-RL pipeline. Results show that activating thinking changes the agent’s tool-use behavior and leads to different Long-Bench QA outcomes. Bold denotes the best score within this block, and underline denotes the runner-up.

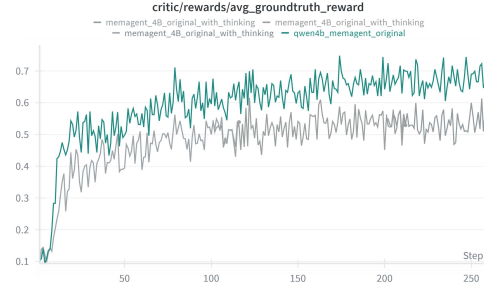


Figure 9: **Training dynamics: thinking vs. no-thinking for MemAgent-RL reproduction.** We plot the training curves of reproduced MemAgent-RL runs on Qwen3-4B with thinking mode enabled and disabled. During training, the no-thinking variant consistently outperforms the thinking variant.

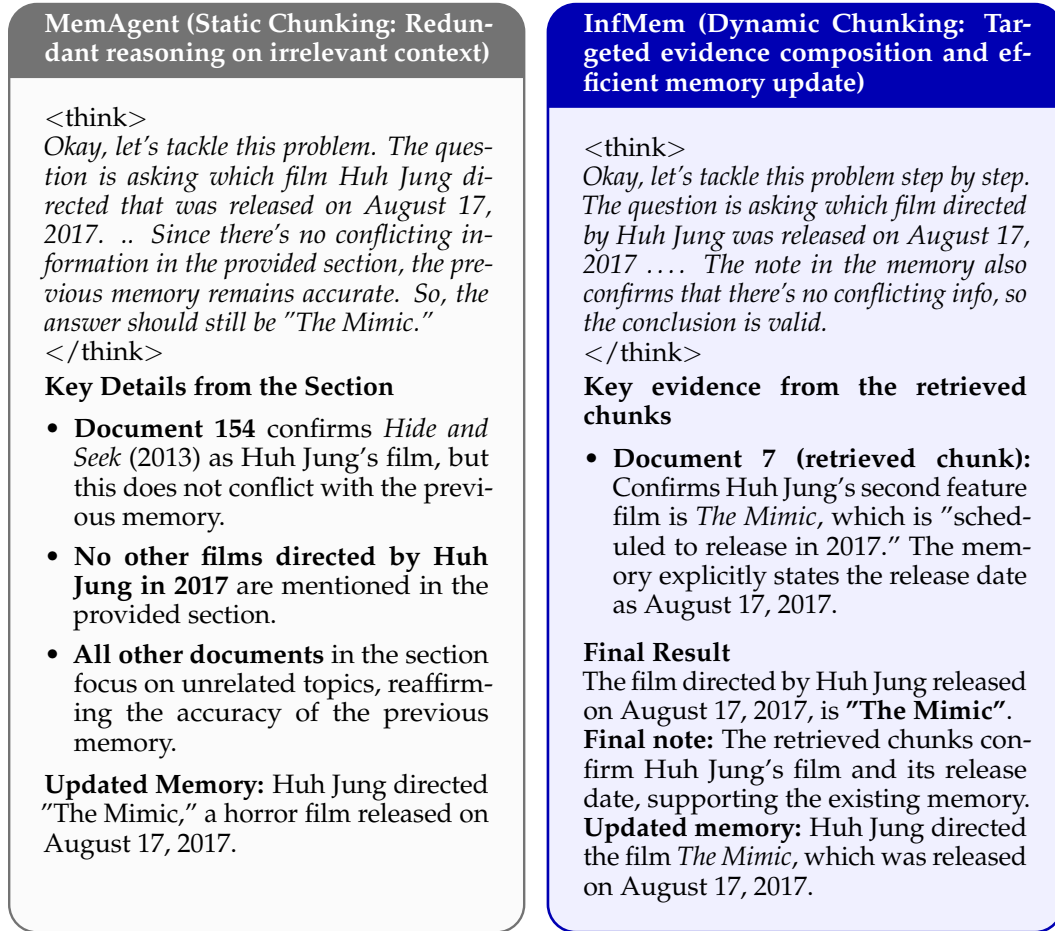


Figure 10: **Comparison of memory update efficiency between MemAgent and InfMem.** This snapshot is extracted from training logs at step 100. Left: MemAgent suffers from significant token redundancy due to exhaustive reasoning over irrelevant chunks, leading to over-thinking. Right: Our InfMem employs a **Dynamic Chunking** strategy within the PreThink-Retrieve-Write protocol, allowing the model to concentrate its reasoning capacity on critical evidence and update long-term memory with higher precision and lower computational cost.

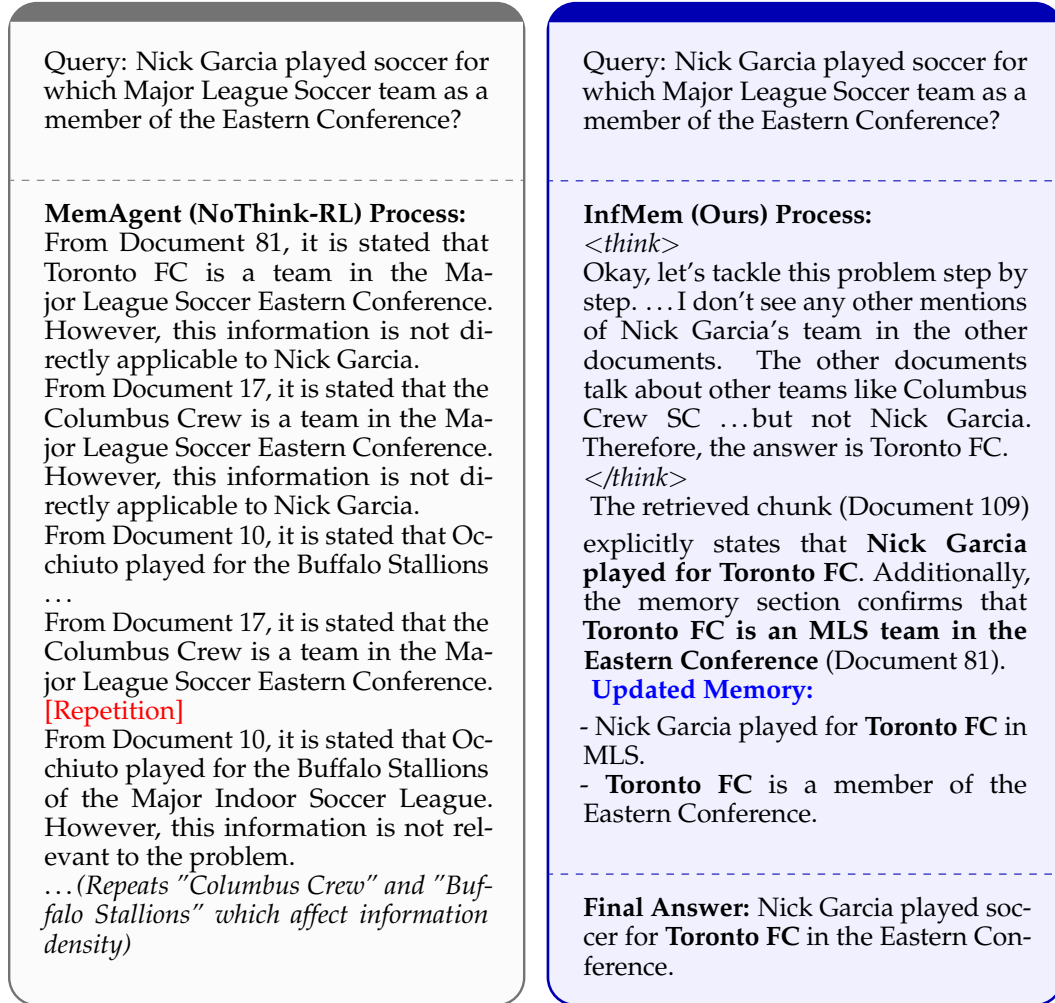


Figure 11: **Qualitative Comparison of Memory Purity.** **Left:** MemAgent (NoThink) is susceptible to recurrent noise, tending to process and accumulate irrelevant information (e.g., details about unrelated teams like Columbus Crew) which dilutes memory utility. **Right:** In contrast, InfMem utilizes the reasoning mechanism to actively filter out these distractors. By synthesizing only the critical evidence, InfMem maintains a memory of **significantly higher quality and purity**, ensuring that only high-fidelity facts relevant to the query are preserved.