INFMEM : LEARNING SYSTEM-2 MEMORY CONTROL FOR LONG-CONTEXT AGENT

Xinyu Wang[1,*] **Mingze Li**[2,*] **Peng Lu**[3] **Xiao-Wen Chang**[1] **Lifeng Shang**[2]
Jinpeng Li[2] **Fei Mi**[2] **Prasanna Parthasarathi**[2] **Yufei Cui**[2]

[1]McGill University [2]Noah's Ark Lab [3]Université de Montréal
xinyu.wang5@mail.mcgill.ca yufei.cui@huawei.com
[*]Equal contribution

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, MemAgent 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**× on average (up to 5.1×) via adaptive early stopping.
Code is available at https://github.com/UCMP13753/InfMem.

1 INTRODUCTION

Long-document question answering increasingly demands reasoning over *extreme-length* contexts under a *bounded* compute/memory budget. In this regime, decisive evidence is often *sparse* and *widely scattered*, thus requires *cross-chunk composition*—e.g., linking an early definition to a later exception clause or reconciling a claim with a delayed numerical qualifier (Liu et al., 2024; Bai et al., 2024). Such settings arise routinely in rigorous synthesis (legal review, technical analysis, codebase reasoning), where correct answers hinge on a few delayed, low-salience facts rather than the global gist (Shaham et al., 2022; An et al., 2024). This creates a *fidelity dilemma*: aggressive segment-wise compression can erase the subtle links needed for later composition, while naively expanding the raw context dilutes attention and buries decisive facts in noise (Weston & Sukhbaatar, 2023; Xu et al., 2023). Resolving this dilemma requires *task-conditioned evidence management*—prioritizing and resurfacing the few *bridging facts and links* that enable multi-hop synthesis under a fixed budget (Chen et al., 2023).

Prior work improves long-context capability via length extrapolation (Press et al., 2021; Su et al., 2024; Peng et al., 2024) and efficient sequence modeling (Liu et al., 2023; Yang et al., 2024b; Gu & Dao, 2024), but largely focuses on capacity rather than organizing evidence for multi-hop reasoning over extreme-length documents. Retrieval-augmented generation (RAG) Lewis et al. (2020b) can surface relevant snippets, yet the resulting evidence is often fragmented and not consolidated into a compact working substrate (Asai et al., 2024; Barnett et al., 2024; Ma et al., 2025). Conversely, bounded-memory agents such as MemAgent offer a bounded-cost profile with a constant-size memory state and single-pass processing, which yields $\mathcal{O}(1)$ memory and $\mathcal{O}(n)$ computation over a document of n segments. However, these agents rely on passive, reactive update policies and are unable to revisit earlier context to recover missing evidence when needed (Packer et al., 2023; Yu et al., 2025).

An ideal state-dependent controller is expected to be capable to decide when evidence is insufficient, what to retrieve, and how to write selectively under a fixed memory budget (Jiang et al., 2023). However, existing approaches lack such a state-dependent controller. We argue that effective bounded-memory long-context processing requires a shift from passive, segment-wise compression to **System-2-style cognitive control** (Kahneman, 2011). Inspired by dual-process accounts of human cognition, we use "System-2" as a *computational* abstraction for explicit, task-conditioned, state-dependent control over *memory operations* (Sumers et al., 2023). From this perspective, long-context reasoning under bounded memory is a *multi-stage* control loop with an explicit *intermediate state*—tracking what is supported, what remains missing for the question, and where to fetch evidence—rather than a single-pass summary of each segment (Wei et al., 2022; Yao et al., 2022). In contrast, many existing bounded-memory agents are largely *System-1-leaning*, relying on reactive heuristics that can work in routine settings but can struggle on multi-hop queries that require non-monotonic evidence access and selective retention (Yu et al., 2025). Concretely, System-2 control instantiates a *monitor–seek–update–stop* loop: (i) monitor whether the current memory suffices for the question, (ii) seek missing support via targeted in-document retrieval, (iii) update the bounded memory to retain question-relevant bridging links under an overwrite budget, and (iv) stop early once sufficient evidence is secured to avoid redundant iterations (Sumers et al., 2023).

To instantiate this System-2-style control, we propose **InfMem**, a long-context agent that executes a structured PRETHINK–RETRIEVE–WRITE protocol with early stopping. At each step, PRETHINK monitors the current memory to assess whether it already suffices to answer the question; if not, it synthesizes a question-conditioned retrieval query and predicts a retrieve size. Whenever PRETHINK chooses to continue (i.e., outputs RETRIEVE rather than STOP), RETRIEVE issues targeted queries over the *entire document*, enabling non-monotonic access to relevant segments. This allows the agent to revisit earlier portions when needed and to check later sections to fill in missing support. WRITE then *jointly* integrates the current segment with retrieved evidence into a bounded overwrite memory, prioritizing the facts and links required for downstream composition under a fixed budget. Finally, **InfMem** applies *early stopping*: once sufficient evidence has been consolidated in memory, it terminates the retrieve–write loop, reducing redundant retrieval and inference steps while avoiding unnecessary overwrites.

Such control is not plug-and-play: protocol design alone does not guarantee reliable retrieve/write/stop decisions. We therefore adopt a practical training recipe, warm-starting **InfMem** with supervised fine-tuning on reasoning-correct trajectories and then applying verifier-based reinforcement learning to align retrieval, writing, and stopping with end-task correctness and efficiency under ultra-long contexts.

Contributions:

- **InfMem: a control-centric agent for long-context QA.** We propose **InfMem**, a bounded-memory agent that employs a PRETHINK–RETRIEVE–WRITE loop to actively *retrieve* missing evidence, *consolidate* memory updates, and *stop early* under fixed budgets.
- **A practical recipe for learning long-horizon control.** We introduce a verifiable SFT→RL pipeline that robustly aligns discrete control decisions (retrieval, writing, and stopping) with long-horizon reasoning rewards.
- **Robust gains with lower inference cost.** On 1M-token benchmarks, InfMem outperforms MemAgent by over **10 points** across Qwen series while reducing inference latency by **3.9**× via adaptive early stopping.

2 RELATED WORK

Long-Context Modeling and Efficiency. Recent advancements have dramatically expanded context windows, with frontier models scaling to million-token regimes (Qwen Team, 2025; Wan et al., 2025b; Yang et al., 2025b; Wan et al., 2025a) and efficient architectures (e.g., linear attention, SSMs like Mamba) reducing the quadratic complexity of self-attention (Gu & Dao, 2024; Yang et al., 2024b). While these methods improve *capacity*, simply fitting more text into the window does not guarantee effective reasoning: performance often degrades on retrieval-heavy tasks due to "lost-in-the-middle" phenomena (Liu et al., 2024; Weston & Sukhbaatar, 2023). Furthermore, monolithic processing of ultra-long documents lacks explicit control over evidence selection. Our work targets this gap by focusing on *active evidence management* under bounded budgets rather than raw architectural capacity.

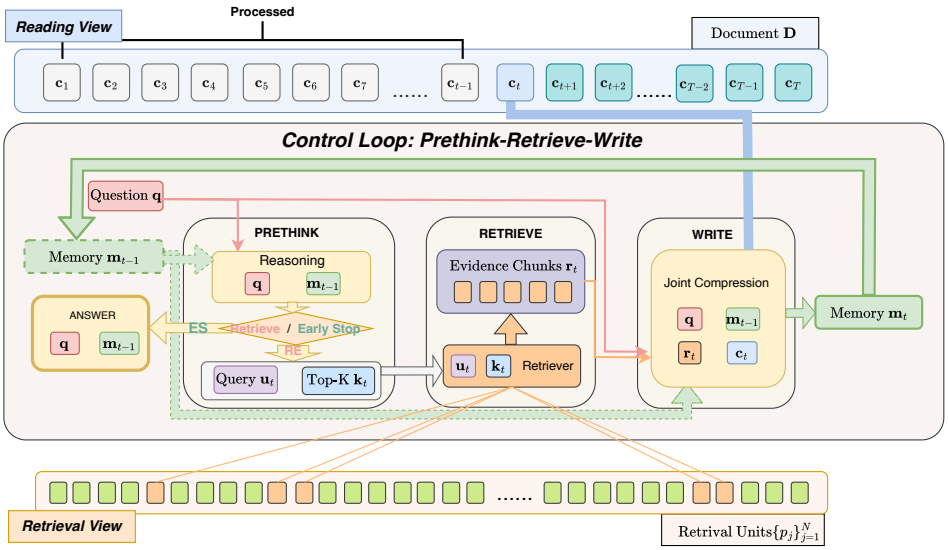

Figure 1: **The InfMem System-2 Framework.** Unlike passive streaming agents, InfMem instantiates an active **System-2 control loop** (PRETHINK–RETRIEVE–WRITE) to manage bounded memory. (1) **PRETHINK** acts as a cognitive controller, *monitoring* memory sufficiency to decide whether to answer immediately (*Early Stop*) or *seek* more information. (2) **RETRIEVE** executes targeted global search, fetching sparse evidence $\mathbf{r}_t$ from the index $\{p_j\}$ to bridge logical gaps. (3) **WRITE** performs *joint compression*, synthesizing the retrieved evidence with the current stream $\mathbf{c}_t$ to update the memory $\mathbf{m}_t$. This loop enables the agent to actively maintain evidence fidelity under extreme context lengths.

Learning-based Memory Controllers. Several recent works explore training models to actively manage memory states. Foundational research by Zhang et al. (2023) formulates LLMs as semi-parametric RL agents that learn to retrieve and update memory. Building on this, approaches such as MEM1 (Zhou et al., 2025), Memory-R1 (Yan et al., 2025), and MemGPT (Packer et al., 2023) introduce specific mechanisms for memory management. However, these methods predominantly target *interactive* or *conversational* settings (e.g., LoCoMo (Maharana et al., 2024)), prioritizing state tracking or persona consistency across indefinite turns. Narrowing the scope to *long-document QA*, MemAgent (Yu et al., 2025) is the most relevant baseline. In contrast to MemAgent's passive updates which risk discarding sparse evidence, InfMem employs a System-2 loop to actively *retrieve* and *consolidate* bridging facts specifically for reasoning.

3 INFMEM FRAMEWORK

InfMem is a bounded-memory agent for long-document question answering that executes an explicit *PreThink–Retrieve–Write* control loop with *early stop*. It reads the document in a single pass, maintains a fixed-size overwrite memory, and decides when the accumulated memory is sufficient to answer the question. When the memory is insufficient, it retrieves additional evidence *from within the same document* and updates the memory by reasoning over the incoming segment together with the retrieved evidence. Early stopping terminates processing once sufficient evidence has been consolidated, reducing redundant updates and inference time.

3.1 STREAMING SETTING AND REPRESENTATIONS

Problem Setting. We consider question answering over a long document in a document-available, single-pass setting. Given a question q and a document D, the goal is to produce an answer $\hat{y}$ using evidence distributed throughout D. Due to the limited context window size and computational constraints, it could be infeasible to feed the entire ultra-long document into an LLM. Therefore, following the scalable streaming formulation popularized by MemAgent (Yu et al., 2025), we

sequentially process the document under a fixed per-step budget using a bounded overwrite memory state.

Streaming chunks and bounded memory. We segment the document D into an ordered stream of T coarse *streaming chunks* $\{c_t\}_{t=1}^{T}$. **InfMem** maintains a bounded memory state m_t (a token sequence) with a fixed budget $|m_t| \leq M$. After reading each chunk, the agent updates its memory by selectively overwriting an older entry, keeping the per-step context size constant and ensuring end-to-end complexity linear in T.

Fine-grained Indexing for Global Access. 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 jump to 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.

3.2 CONTROL LOOP: PRETHINK–RETRIEVE–WRITE WITH EARLY STOP

As illustrated in Figure 1, **InfMem** views an ultra-long document not as a monolithic block but as a controlled stream of evidence under a fixed context budget. The model maintains a compact memory m_t as ordinary tokens inside the LLM context window, so the base LLM architecture and generation process remain unchanged. A key challenge is that blindly overwriting memory after each chunk can discard low-salient but composition-critical evidence needed for multi-hop reasoning. To tackle this challenge, **InfMem** propose to decouple **planning** from **evidence-aware writing** and use global in-document retrieval over fine-grained units to *shape memory updates* (Table 3a in Appx. A.4.1).

Step protocol (monitor–seek–update–stop). At step t, **InfMem** treats the bounded memory m_{t-1} as the intermediate state. PRETHINK conditioned only on (q, m_{t-1}) is first run to **monitor** whether the current memory is sufficient to answer q. If sufficient, the agent outputs "STOP" and terminates early. Otherwise, it outputs "RETRIEVE", and then accordingly synthesizes a single retrieval query, predicts how many retrieval units to fetch, and invokes RETRIEVE to **seek** sparse evidence globally from the same document, producing a compact retrieved context r_t. Finally, WRITE **updates** the memory by reasoning over the incoming chunk c_t together with r_t and overwriting the memory via bounded *joint* compression under the fixed budget.

PRETHINK: the explicit controller. 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 $a_t \in \{\text{"STOP"}, \text{"RETRIEVE"}\}$: whether the current memory is sufficient to answer q (stop) or additional in-document evidence is needed (retrieve);
- QUERY u_t (if $a_t =$ "RETRIEVE"): a single dynamic query synthesized from (q, m_{t-1});
- TOPK $k_t \in \{1, \ldots, K_{\max}\}$ (if $a_t =$ "RETRIEVE"): 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 retrieve* and *how much to retrieve*. 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: global in-document evidence. If $a_t =$ "RETRIEVE", **InfMem** retrieves top-k_t relevant retrieval units from the same document (no external corpus) and concatenates them into a compact context:

$$\begin{aligned} P_t &\leftarrow \textsc{Retrieve}(u_t, k_t; \{p_1, \ldots, p_N\}), \\ r_t &\leftarrow \mathrm{Concat}(P_t), \end{aligned} \tag{1}$$

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

WRITE: evidence-aware composition and joint compression. If $a_t =$ "RETRIEVE", **InfMem** overwrites the memory with a bounded new state:

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

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 and end-of-sequence answering. If $a_t =$ "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:

$$\hat{y} \leftarrow \textsc{Answer}(q, m_\star),$$

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

4 TRAINING INFMEM

InfMem instantiates explicit (System-2-style) control over a bounded-memory stream via the PRETHINK–RETRIEVE–WRITE loop with EARLY STOP (Section 3.2). We post-train a LLM as the base model to produce protocol-valid intermediate outputs (e.g., **structured decision tuples** and **compressed memory states**) and to learn long-horizon policies (retrieve/write/stop) under delayed feedback using two stages: (1) **SFT warmup** for protocol adherence, and (2) **RL alignment** for task success and efficiency.

4.1 SFT WARMUP VIA SUPERVISED DISTILLATION

Train–test consistent prompting. We distill a strong teacher model (e.g., `Qwen3-32B`) to a smaller student model using prompt templates that strictly mirror the inference-time PRETHINK–RETRIEVE–WRITE loop with early stopping. Each SFT trajectory follows the inference-time loop: PRETHINK first outputs an action $a_t \in \{$"STOP", "RETRIEVE"$\}$. If $a_t =$ "RETRIEVE", the teacher executes RETRIEVE and then WRITE; if $a_t =$ "STOP", the rollout terminates and the final answer is produced. This enforces strict *train–test consistency*: the student receives supervision signals only on inference-valid actions, emphasizing on protocol format and execution reliability rather than task-specific specialization.

Across tasks, the teacher receives the question and executes the protocol till the termination. We utilize the golden question decompositions or supporting evidence provided in the training sets as high-level hints to guide the synthesis of planning traces. During evaluation, we refrain from disclosing any such auxiliary information to the LLM or the agent system. All prompt templates and formatting details are provided in § A.2.

Data filtering and supervised objective. We construct a warmup-set from QA tasks to demonstrate evidence aggregation and iterative memory updates (Appendix A.3.1). Only trajectories, whose final answer is correct under the official protocol ($\mathrm{EM}(\hat{y}, y) = 1$), are retained. The string/regex filters are applied to remove any ground-truth leakage.

Each rollout from the teacher is serialized into a single protocol-formatted dialogue τ. The student is trained with masked next-token prediction purely on *agent response tokens* (masking all system/user/prompt tokens). Let $\mathcal{Y}(\tau)$ index the response tokens in τ, with $\mathrm{prefix}_i(\tau)$ denoting all preceding tokens. The objective is:

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

The gradients would be backpropagated through all realized steps up to the teacher's termination, which jointly supervise the protocol control records, the bounded memory updates, and the final answers.

Table 1: **Cross-model and ultra-long QA results up to 1M tokens.** We compare YaRN, RAG-top6, MemAgent, and InfMem across **Qwen3-1.7B/4B** and **Qwen2.5-7B** on synthesized RULER-style benchmarks under increasing context lengths. Both MemAgent and InfMem provide consistent train-free gains over long-context baselines, and RL further amplifies the improvements.

Metric	Qwen3-1.7B						Qwen3-4B						Qwen2.5-7B					
	Framework				+RL		Framework				+RL		Framework				+RL	
	YaRN	RAG top6	MemAgent	InfMem	MemAgent	InfMem	YaRN	RAG top6	MemAgent	InfMem	MemAgent	InfMem	YaRN	RAG top6	MemAgent	InfMem	MemAgent	InfMem
avg	13.38	18.50	20.18	**37.71**	40.67	**50.84**	25.45	26.05	43.61	**50.25**	54.56	**66.40**	21.41	19.77	37.06	**47.73**	52.07	**60.30**
HQA																		
28k	22.30	33.49	28.52	**47.84**	**59.71**	56.80	50.77	48.46	52.55	**59.73**	71.18	**71.44**	35.70	33.51	44.96	**45.70**	**65.58**	59.20
56k	17.86	32.09	31.47	**47.81**	**53.45**	52.59	42.07	43.69	51.27	**58.69**	66.21	**68.73**	31.74	29.87	45.93	**48.56**	**62.88**	62.23
112k	17.57	30.35	30.16	**41.98**	49.91	**56.59**	35.19	42.52	44.02	**51.33**	62.42	**71.24**	25.42	31.45	42.76	**47.98**	**61.55**	57.75
224k	10.58	18.83	19.95	**44.94**	49.12	**56.63**	10.96	24.11	44.68	**47.82**	59.12	**67.42**	13.93	16.94	34.77	**49.65**	59.95	**60.55**
448k	5.42	12.83	20.23	**43.04**	44.17	**51.46**	8.34	13.27	40.47	**51.71**	58.84	**67.75**	9.23	8.61	33.07	**46.70**	57.09	**62.34**
896k	2.91	4.91	18.92	**41.62**	42.50	**51.31**	5.26	3.73	40.03	**49.07**	51.70	**66.13**	3.91	2.39	34.47	**42.60**	**58.39**	57.51
SQuAD																		
32k	20.89	41.13	25.33	**57.95**	50.91	**59.30**	48.70	55.66	53.82	**65.75**	**69.49**	65.31	34.36	36.80	45.02	**55.77**	61.95	**61.70**
64k	14.28	31.76	26.59	**51.41**	48.77	**55.68**	39.80	49.91	54.73	**61.07**	**69.84**	66.42	31.55	33.55	47.06	**53.98**	57.94	**64.19**
128k	16.44	29.39	30.73	**56.73**	49.18	**56.33**	36.79	45.05	51.80	**64.17**	**72.96**	66.05	29.44	27.93	49.15	**54.23**	58.26	**58.82**
256k	18.18	22.03	24.05	**50.82**	48.50	**53.84**	24.89	35.76	46.23	**59.23**	**71.24**	63.53	27.50	22.15	41.83	**53.03**	53.23	**61.71**
512k	18.86	20.62	33.45	**49.27**	54.48	**58.24**	34.93	26.36	51.62	**64.99**	77.21	**78.12**	20.23	20.70	50.92	**63.08**	**69.85**	69.27
1M	4.32	2.56	25.20	**48.09**	47.29	**59.56**	9.63	5.26	48.91	**59.38**	**77.74**	73.81	4.59	2.57	44.97	**55.99**	**68.63**	67.71
MuSiQue																		
32k	12.27	13.84	14.51	**22.86**	30.50	**43.76**	19.65	19.84	29.02	**41.35**	41.79	**56.58**	19.93	18.37	31.45	**37.09**	36.67	**46.27**
64k	8.22	6.12	13.91	**23.03**	31.37	**40.95**	14.00	11.94	34.03	**38.06**	41.55	**57.19**	17.27	14.13	26.29	**36.73**	32.82	**46.05**
128k	10.94	8.73	7.52	**21.31**	30.41	**42.67**	9.48	15.78	28.23	**35.31**	36.62	**55.62**	10.61	10.40	20.71	**41.33**	37.79	**48.13**
256k	7.77	6.85	7.70	**24.03**	26.89	**43.48**	14.52	13.75	25.50	**38.04**	43.04	**61.39**	12.79	10.86	25.52	**38.79**	44.91	**57.49**
512k	9.62	5.32	12.54	**17.75**	21.03	**41.81**	7.48	7.97	**32.93**	31.57	35.64	**59.59**	12.94	10.12	21.49	**40.14**	35.77	**55.26**
1M	4.51	2.75	10.27	**24.90**	24.05	**38.18**	8.30	3.80	25.62	**34.20**	35.91	**56.86**	3.15	2.55	21.77	**41.49**	38.40	**58.57**
2Wiki																		
32k	16.45	26.91	16.92	**33.52**	40.90	**56.12**	49.71	39.58	**55.62**	54.70	56.43	**70.66**	37.41	39.20	44.71	**44.52**	49.57	**68.78**
64k	17.08	26.70	20.57	**32.88**	39.55	**45.98**	40.86	28.34	47.56	**49.88**	48.55	**74.84**	42.00	37.38	40.06	**49.53**	51.68	**64.80**
128k	19.27	28.18	22.13	**32.83**	34.28	**51.68**	39.72	30.47	**50.27**	47.40	46.18	**70.46**	33.51	29.75	46.31	**50.6**	49.66	**61.88**
256k	17.00	15.65	11.72	**31.92**	34.09	**50.38**	20.39	20.02	43.99	**46.20**	41.42	**66.62**	21.67	13.96	34.85	**48.15**	47.73	**63.55**
512k	14.77	15.44	13.37	**27.28**	32.54	**48.51**	20.16	22.52	48.23	**50.81**	39.09	**71.54**	22.77	15.66	32.93	**54.74**	46.31	**65.19**
1M	13.64	7.47	18.62	**31.16**	32.52	**48.34**	19.12	17.30	45.45	**45.55**	35.18	**66.39**	12.15	5.60	28.49	**50.62**	43.18	**68.20**

Role of SFT warmup. In practice, the warmup stage mainly instructs the mechanics of the PRETHINK–RETRIEVE–WRITE protocol—emitting valid retrieve calls, producing well-formed bounded-memory updates, generating final answers, and executing early stopping. It will not instruct the System-2 *control policy—when* to stop, and when to continue, *what/how much* evidence to retrieve and *what* to write under the overwrite budget. These functions will be learned in the subsequent RL stage under delayed outcomes.

4.2 RL ALIGNMENT WITH REWARD DESIGN

While the warmup with SFT ensures the protocol-compliant execution, it neither learns the System-2 control policy under delayed feedback—*when* to stop, and when to continue, *what/how much* to retrieve and *how* to write—nor does it robustly align these decisions with end-task success. Therefore, we apply RL with the outcome-based rewards to align the task success, protocol soundness, and efficient early stopping.

Multi-conversation GRPO backbone. We follow the paradigm of multi-conversation GRPO/-DAPO in MemAgent for agentic long-context workflows (Yu et al., 2025). Each rollout contains multiple memory-update *steps* (turns) and a final answering step, while the final outcome reward is shared across all preceding steps to enable long-horizon credit assignment (Yu et al., 2025). For each query, we sample a group of G rollouts with outcome rewards $\{R_i\}_{i=1}^{G}$ and compute the corresponding advantages as follows:

$$\bar{R} = \frac{1}{G}\sum_{i=1}^{G} R_i, \qquad \hat{A}_i = R_i - \bar{R}. \tag{4}$$

With the advantages, the clipped surrogate objective is optimized with KL regularization:

$$J(\theta) = \mathbb{E}_{i,t}\Big[\min\Big(r_{i,t}(\theta)\hat{A}_i,\ \mathrm{clip}(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_i\Big) - \beta D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid s_{i,t}) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid s_{i,t})\big)\Big]. \tag{5}$$

where $r_{i,t}(\theta) = \pi_\theta(a_{i,t} \mid s_{i,t})/\pi_{\theta_{\mathrm{old}}}(a_{i,t} \mid s_{i,t})$; t indexes the order of tokens in the concatenated rollout trajectory (including tool-call and memory-writing tokens), while the reward components below are defined at the rollout level. We omit advantage scaling with std for stability when rewards are sparse and near-binary. Hyperparameters are provided in Experiments A.1.

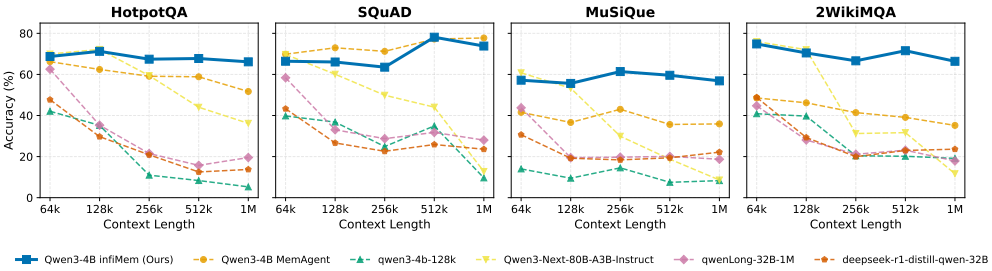

Figure 2: **Long-context scaling of Qwen3-4B up to 1M tokens on synthesized long-context QA benchmarks.** InfMem demonstrates remarkable robustness in long-context scaling, maintaining consistent accuracy on synthetic benchmarks up to 1M tokens without performance degradation

Protocol-soundness verifiers. To keep exploration within protocol-valid regions and prevent invalid intermediate outputs from disrupting downstream steps, we add two binary rollout-level verifiers:

- **Function-call verifier** R_{call}: equals 1 if and only if all function calls are well-formed and parsable; otherwise is set to 0.
- **Memory verifier** R_{mem}: equals 1 if and only if every memory-update step outputs a complete UPDATEDMEMORY field that is not truncated and respects the fixed memory budget; otherwise:0.

The exact verifier definitions are provided in Appx. A.3.2.

Final task reward. To optimize the end task, we define a rule-based ground-truth reward computed from the final predicted answer:

$$R_{\text{gt}}(\hat{y}, y) = \mathbf{1}\{\text{equiv}(\hat{y}, y)\}, \tag{6}$$

where $\text{equiv}(\cdot, \cdot)$ follows the official benchmark evaluation protocol (e.g., exact-match normalization).

Early-stop shaping. We add an InfMem-specific shaping term that rewards stopping soon after the memory first becomes sufficient to answer. Let t_{first} be the earliest memory-update step at which the question can be answered correctly using *only* the current memory (EM=1 under the official normalization, evaluated by a frozen answer-only evaluator), and let t_{stop} be the agent's stopping step. Define $d = t_{\text{stop}} - t_{\text{first}}$ (so $d = 1$ stops immediately after sufficiency) and assign

$$R_{\text{early}} = \gamma^{\,d-1}, \quad \gamma \in (0, 1), \tag{7}$$

so $R_{\text{early}} = 1$ when the agent stops immediately after the first sufficient-memory step, to prevent redundant overwrites.

Final outcome reward. The outcome reward R_i used in Eq. (4) is a weighted combination of the above components:

$$R = \sum_{w} \alpha_w R_w, \text{ s.t. } w \in \{\text{gt, early, call, mem}\}, \tag{8}$$

where all coefficients are specified in the experiments.

5 EXPERIMENT SETUP

5.1 DATASETS

We utilize four datasets spanning a spectrum of reasoning demands: **SQuAD** (Rajpurkar et al., 2016) for single-hop extraction, and **HotpotQA** (Yang et al., 2018), **2WikiMultiHopQA** (Ho et al., 2020), and **MuSiQue** (Trivedi et al., 2022) for complex multi-hop aggregation across documents. These corpora form the basis for constructing our synthetic long-context training data and evaluation benchmarks, as detailed in §A.3.

LongBench. We additionally 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. This provides an external reference point for long-context QA performance and complements our controlled settings.

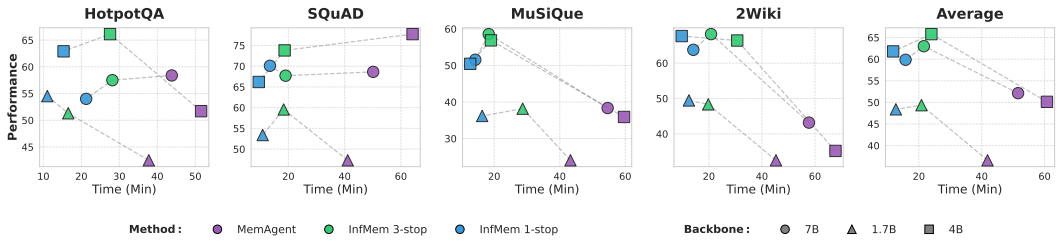

Figure 3: **Inference Efficiency versus QA Performance on 1M Context Scaling.** Notably, InfMem exhibits exceptional proficiency in long-range multi-hop reasoning, preserving high-fidelity performance without the computational overhead typically associated with extreme sequence lengths.

5.2 Baselines

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.

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

We compare InfMem against three distinct categories of long-context baselines: (1) Length Extrapolation: The official train-free YaRN (Peng et al., 2024) setting; (2) Retrieval Augmentation: A standard RAG (Lewis et al., 2020a) pipeline; (3) Agentic Memory System: MemAgent (Yu et al., 2025). Additionally, we reference high-capacity models (e.g., Qwen3-Next-80B-A3B-Instruct (Qwen Team, 2025), DeepSeek-R1-Distill-Qwen-32B (DeepSeek-AI, 2025), and QwenLong-L1-32B (Wan et al., 2025a)) to contextualize performance limits with disparate compute budgets.

6 Empirical Results

6.1 Cross-backbone results up to 1M tokens

Table 1 reports results on synthesized long-context QA benchmarks evaluated across Qwen3-1.7B/4B and Qwen2.5-7B.We observe that standard long-context baselines degrade sharply in the ultra-long regime.Specifically, YaRN exhibits a distinct performance cliff beyond 128k tokens, with accuracy often collapsing to single digits at the 1M mark (e.g., dropping to $\sim$4% on Qwen2.5-7B). Similarly, RAG performance decays as information density decreases, struggling to locate decisive evidence when it is widely dispersed across million-token contexts.

Among memory-based approaches, InfMem consistently achieves the strongest performance. While MemAgent remains competitive on tasks with simpler evidence retrieval patterns (e.g., SQuAD), it lags substantially on complex multi-hop benchmarks such as MuSiQue and 2WikiMultiHopQA. This divergence suggests that the recurrent, reactive compression of MemAgent is more prone to gradual information loss over long horizons, whereas InfMem's architecture better preserves long-range dependencies. Finally, the proposed SFT→RL training recipe yields consistent gains by optimizing the agent's decision-making process. Consequently, RL-InfMem establishes a decisive lead, outperforming RL-MemAgent by an average margin of over 10% across the evaluated backbones.

6.2 Scaling behavior with increasing context length

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

6.3 Transfer to LongBench QA

Crucially, these gains are not confined to our synthesized ultra-long setting. As shown in Table 2, performance improvements transfer to LongBench QA, which features shorter contexts with higher information density and thus places greater emphasis on evidence analysis and selection rather than merely preserving memory over long horizons (detail explanation in §D.2). Across backbones, InfMem consistently outperforms MemAgent in both train-free and RL-enhanced settings, while RL further widens the gap over YaRN. Overall, the results suggest that InfMem improves not only robustness under extreme length (up to 1M tokens) but also the quality of reasoning-oriented evidence management on standard long-context QA benchmarks.

6.4 Early stopping

Early stopping is key to making recurrent retrieval scalable. Figure 3 illustrates the efficiency–quality trade-off on 1M-token tasks. Across Qwen3-1.7B/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 **5.1**×, **3.3**×, and **3.3**×. 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.

6.5 Further Analysis and Ablation Study

Beyond the main results, we also provide comprehensive ablation and further studies in Appx. C, including the retrieval chunk size selection, analysis of early stop, ablation on thinking mode and the analysis of memory retention.

7 Conclusion

In this work, we present InfMem, a cognitive agent designed to resolve the fidelity dilemma in ultra-long context reasoning through a System-2 paradigm. 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× 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".

REFERENCES

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.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pp. 194–199, 2024.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*, 2023.

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.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023.

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*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2020b. URL https://arxiv.org/abs/2005.11401.

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 Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oFBu7qaZpS.

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, 2024. URL https://openreview.net/forum?id=wHBfxhZu1u.

Ofir Press, Noah A Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

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.

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.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multi-hop 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-l1: Towards long-context large reasoning models with reinforcement learning. *CoRR*, abs/2505.17667, 2025a. doi: 10.48550/ARXIV.2505.17667. URL https://doi.org/10.48550/arXiv.2505.17667.

Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. Qwenlong-l1: Towards long-context large reasoning models with reinforcement learning, 2025b. URL https://arxiv.org/abs/2505.17667.

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.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025*, 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.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent, 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.

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 PIPELINE

Backbones. 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.

Data Preparation. Following the protocol-valid trajectories in §4, we implement the pipeline as follows: (1) SFT Mixtures: We pack synthetic trajectories from HotpotQA, SQuAD, and MuSiQue (§A.3.1) into 32k token sequences for efficiency. (2) Teacher Model: We employ Qwen3-32B to generate PRETHINK–RETRIEVE–WRITE traces for distillation. (3) RL Samples: We utilize a long-context variant of HotpotQA (§A.3.2) to provide dense signals for multi-hop reasoning.

Stage 1: SFT Warmup. We use a learning rate of 4.0×10^{-5} with a cosine learning rate scheduler and a global batch size of 256. The training duration is tailored to the base model's capabilities: for the Qwen3-1.7B and 4B (already reasoning-optimized) backbones, we train for 1 epoch to adapt to the protocol trajectories. For Qwen2.5-7B-Instruct, which is a general-purpose model, we extend the training to 4 epochs to ensure it effectively masters the underlying reasoning paradigm.

Stage 2: RL Alignment. Starting from the SFT checkpoints, we apply GRPO with $G = 4$ rollouts per prompt. The sampling temperature is set to 1.0 with top-p $= 1.0$. We use a KL divergence coefficient $\beta = 0.001$. The optimization is conducted with a training batch size of 128 (mini-batch size of 8) and a constant learning rate of 1×10^{-6}.

A.2 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 4) 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 5) 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.3 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.3.1 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.

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.3.2 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.3.3 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, 2WikiMultihopQA, and MuSiQue; we also include the synthetic SQuAD setting used in MemAgent for direct comparison. We use each dataset's

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

[2] In 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.

Retrieval Planner Prompt Template

Template

```
You are a Retrieval Planner.
Your ONLY task is to decide whether to perform another retrieval using `retrievesearch`,
or STOP retrieving.
You MUST NOT answer the QUESTION.
Another model will use MEMORY to answer later.
Guidelines:
- Retrieval is cheap.  Unless MEMORY clearly contains all essential information, you are
encouraged to retrieve.
- You may retrieve multiple times.  At each step, refine your search direction.
- Avoid repeating any previous queries in RETRIEVAL_HISTORY (unless meaningfully
refined).
When deciding whether to retrieve again:
1.  Break the QUESTION into specific sub-questions or information needs.
2.  Compare these needs with what MEMORY already contains.
3.  Identify which facts are still missing, uncertain, or incomplete.
4.  If something important is missing, design a NEW search query focused only on that
missing information.
 - You may explore related clues hinted in MEMORY.
 - Queries should be concise, specific, and actionable.
5.  If MEMORY already contains all necessary information, choose to STOP.
If you choose retrieval, you MUST output a function call to `retrievesearch` with:
- a new `query` (different from RETRIEVAL_HISTORY unless refined),
- and a `top_k` suited to your confidence (small:  focused; large:  broad exploration).
- In early retrieval steps, you may exlore more documents.
- In later steps, focus on refining MEMORY.
ORIGINAL QUESTION:
{prompt}
<retrieval_history>
{retrieval_history}
</retrieval_history>
CURRENT MEMORY:
{memory}
```

Figure 4: Prompt template for the Retrieval Planner, which decides whether to call `retrievesearch` again or stop, without answering the question.

Memory Update Prompt Template

Template

```
You are presented with a problem, a section of an article that may contain the answer
to the problem, and a previous memory.  Please read the provided section carefully and
update the memory with the new information that helps to answer the problem.
The given section has two parts.  One is a retrieved chunk, which is retrieved by the
given question.  Another is the recurrent chunk which is provided recurrently.  Both
chunks might contain useful information, while the retrieved chunk may have a higher
chance.
<problem>
{prompt}
</problem>
<retrieved_chunk>
{retrieve}
</retrieved_chunk>
<recurrent_chunk>
{chunk}
</recurrent_chunk>
<memory>
{memory}
</memory>
Updated memory:
```

Figure 5: Prompt template for memory updating, integrating both retrieved and recurrent chunks to refine the memory state.

test split and sample 128 instances per task. 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

No-think (question-to-query copying)

Original question & correct answer
Q: Which state does Sang-Wook Cheong work in as a materials scientist?
A: New Jersey.

Previous retrieval history
Step 1: query='Sang-Wook Cheong materials scientist state', top_k=5
Step 2: query='Sang-Wook Cheong materials scientist state', top_k=5
Step 3: query='Sang-Wook Cheong materials scientist state', top_k=5

Function call
FUNCTION: retrievesearch
ARGS: {"query": "Sang-Wook Cheong materials scientist state", "top_k": 5}

Memory update
Updated memory:
Sang-Wook Cheong is a Korean American materials scientist at Rutgers University.
The state where he works is New Jersey.

Think (SFT PRETHINK: plan with history & memory)

Original question & correct answer
Q: Where is the company headquarters that makes the GU230 mobile phone?
A: Yeouido-dong, Seoul, South Korea.

Previous retrieval history
Step 1: query='GU230 mobile phone manufacturer', top_k=5
Step 2: query='GU230 mobile phone company headquarters', top_k=5

Function call (planned follow-up)
<think>
Okay, let's see. The user is asking where the headquarters of the company that makes the GU230 mobile phone is located. The retrieval history shows that two searches have been done so far.
...
next query should focus on confirming the relationship between M.Mobile and the GU230 model, maybe using "M.Mobile GU230 manufacturer" to see if there's a direct link in the documents.
</think>
FUNCTION: retrievesearch
ARGS: {"query": "M.Mobile subsidiaries and mobile phone production", "top_k": 5}

Memory update (evidence composition)
<think>
Okay, let's tackle this problem step by step. The user is asking where
...
does mention LG as the manufacturer. The prior analysis missed that. So the correct answer is the headquarters of LG Electronics as per Document 96.
</think>
Updated memory:
GU230 is manufactured by LG Electronics.
LG Electronics is headquartered in Yeouido-dong, Seoul, South Korea.

Figure 6: **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.

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.4 TRAINING SETUP

A.4.1 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).

Before RL (retrieval strategy preserved, but fails to extract key evidence)

Original question & correct answer
Q: which former lawyer previously held the seat held by Yuh-Line Niou
A: Sheldon Silver

Previous retrieval history
Step 1: query='Yuh-Line Niou seat current occupant', top_k=5
Step 2: query='Yuh-Line Niou seat previous occupants', top_k=5
Step 3: query='Frank Lautenberg career', top_k=5

Function call
FUNCTION: retrievesearch
ARGS: {"query": "Yuh-Line Niou seat previous occupants", "top_k": 5}

Memory update (incorrect composition)
Updated memory:
The seat previously held by Yuh-Line Niou is linked to New Jersey's Senate history.
Frank Lautenberg succeeded a retiring Republican senator in the 1982 New Jersey Senate election.
Thus, Frank Lautenberg is the most plausible former lawyer for the seat.

Answer (wrong)
Frank Lautenberg

After RL (same retrieve pattern, but extracts the key sentence and composes correctly)

Original question & correct answer
Q: which former lawyer previously held the seat held by Yuh-Line Niou
A: Sheldon Silver

Previous retrieval history
Step 1: query='Yuh-Line Niou seat', top_k=5
Step 2: query='seat held by Yuh-Line Niou previous holder', top_k=5
Step 3: query='Yuh-Line Niou seat holder', top_k=5

Function call
FUNCTION: retrievesearch
ARGS: {"query": "seat held by Yuh-Line Niou previous holder", "top_k": 5}

Memory update (key evidence extracted)
Updated memory:
Yuh-Line Niou serves in the New York State Assembly for the 65th district.
The 65th district seat was previously held by **Sheldon Silver**.
Sheldon Silver is a **former lawyer**.

Answer (correct)
Sheldon Silver

Figure 7: **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.

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 §D.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 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 CASE STUDY

B.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 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 4 and Table 5 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

Before RL (uncontrolled exploration, failure to identify the target franchise)

Original question & correct answer
Q: How many original games are there in the tactical role-playing video game franchise developed by the same Japanese company who developed the "Wars" series of games?
A: Twelve

Previous retrieval history
Step 3: query='Intelligent Systems other RPG franchises', top_k=5
Step 4: query='Intelligent Systems other franchises', top_k=3

Function call
<think>
Okay, let's break this down. The user is asking how many original games are in the tactical role-playing franchise
...
But wait, maybe the memory is missing some information? For example, maybe there's a third game. Let's search for other Japanese tactical RPG franchises besides Sakura Wars to validate.
</think>
FUNCTION: retrievesearch
ARGS: {"query": "Japanese tactical RPG franchises besides Sakuno Wars", "top_k": 4}

Memory update (misidentification)
Updated memory:
The "Wars" series is developed by Intelligent Systems.
No other tactical RPG franchise by Intelligent Systems is explicitly identified.
Advance Wars is part of the same series and does not constitute a separate franchise.

Answer (wrong)
The franchise has **1 original game**.

After RL (top-k–aware exploration and correct evidence composition)

Original question & correct answer
Q: How many original games are there in the tactical role-playing video game franchise developed by the same Japanese company who developed the "Wars" series of games?
A: Twelve

Previous retrieval history
Step 3: query='Fire Emblem original game count number', top_k=5
Step 4: query='Fire Emblem series game count', top_k=5

Function call
<think>
Okay, let's see. The user is asking about the number of original games in the tactical role-playing video game franchise
...
The next step would be to search for the exact count, possibly using a query like "Sakura Wars original game count number" again, but maybe with a different top_k. Or maybe check if the franchise's official website or a game database has the count. Let's try another search with a slightly different query to see if any of the retrieved chunks mention the number.
</think>
FUNCTION: retrievesearch
ARGS: {"query": "Sakura Wars series game count", "top_k": 10}

Memory update (key facts extracted)
Updated memory:
The "Wars" series is developed by **Intelligent Systems**.
Fire Emblem is a tactical role-playing franchise developed by Intelligent Systems.
The Fire Emblem series consists of **twelve original games**, along with remakes and spin-offs.

Answer (correct)
The franchise is **Fire Emblem**, which has **twelve original games**.

Figure 8: **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.

of intermediate actions.

Qualitatively, Figure 6 visualizes retrieval trajectories on the same instance: without PRETHINK, 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.

B.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 7, 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

Table 3: 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)
  $(a_t, u_t, k_t) \leftarrow \textsc{PreThink}(q, m_{t-1})$
  if $a_t = \text{STOP}$ **then**
    break // Early stopping triggered
  end if
  // Step 2: Seek (Retrieve)
  if $a_t = \text{RETRIEVE}$ **then**
    $r_t \leftarrow \textsc{Retrieve}(u_t, k_t; \{p_j\})$
  end if
  // Step 3: Update (Write with Joint Compression)
  $m_t \leftarrow \textsc{Write}(q, m_{t-1}, c_t, r_t; M)$
end for

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

(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 *joint compression*, 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

the decisive evidence chain (Niou → NY Assembly 65th district → Sheldon Silver) and writes a compact, answer-ready memory, enabling a correct final answer.

Figure 8 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, 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.

B.3 EARLY STOP

Beyond accuracy, early stopping substantially improves inference efficiency. As shown in Fig. 9, 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

Table 4: **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 5: **Performance gains from Train-free to SFT and RL across model scales.** Δ_{TF} and Δ_{SFT} denote absolute improvements over Train-free and SFT, respectively.

	Train-free		SFT				RL			
Model	avg_LB	avg_RULER	avg_LB	Δ_{TF} (LB)	avg_RULER	Δ_{TF} (RULER)	avg_LB	Δ_{SFT} (LB)	avg_RULER	Δ_{SFT} (RULER)
Qwen3-1.7B	37.06	37.70	35.22	−1.84	43.71	8.49	41.54	+6.31	50.84	+7.12
Qwen2.5-7B	36.85	47.95	43.00	+6.15	49.30	+1.34	46.46	+3.46	59.53	+10.23
Qwen3-4B	46.64	50.20	46.80	+0.16	54.85	+4.60	50.92	+4.12	66.40	+11.55

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.

C Ablation

C.1 Retrieval Chunk Size Selection

We study the effect of retrieval chunk size under a fixed retrieval budget in Table 6. 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 6 reports accuracy on long-context QA benchmarks (HQA 28k, SQuAD 32k, MuSiQue 32k, 2Wiki 32k) as well as LongBench QA.

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.

Table 6: **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 7: **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

C.2 Early Stop Analysis

To further investigate the impact of the stopping policy on the InfMem framework, we provide a comparative analysis between the 1-stop and 3-stop variants. Table 7 summarizes the raw data for the visualization in Figure 3. As observed, the 1-stop variant offers the lowest latency but suffers from performance degradation due to the premature truncation of evidence chains. In contrast, our default 3-stop variant—which is used for all main results in this paper—occupies the Pareto frontier by balancing negligible computational overhead with significantly higher answer accuracy and stability. This confirms that a slightly conservative stopping policy is essential for preserving critical evidence without sacrificing the overall efficiency of the PreThink-Retrieve-Write protocol.

D Ablation and Analysis of Thinking Dynamics

D.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 11), 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 9. 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.

D.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 9, 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 12). A key observation from Table 9 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%).

Table 9: 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 Think-RL	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
	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

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. The main elements here are: ”American composer”, ”music for the musical”, and ”Something More!”. The key is to find the composer associated with this specific musical.
...
Since the memory is empty, the first retrieval is necessary to get any information. The user’s question requires the composer’s name, so the first search should aim to find that. The top_k of 5 is a good starting point to cast a wide net. After that, 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: retrievesearch
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!’. The retrieval history shows that the first search was for ”American composer for Something More!” with top_k=5. The memory from that search didn’t find any relevant info, except mentioning Syren’s album ”Something Has To Separate”.
...
So, the next step is to perform another retrieval with a refined query, perhaps focusing on the band Syren and their album title. That should check if the composer is mentioned in that context. If that doesn’t work, maybe try a different approach, 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: retrievesearch
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 memory still doesn’t have a clear answer, though. The closest they found was the band Syren with an album titled ”Something Has To Separate,” but that’s not the same as ”Something More!”.
...
The user’s question is about ”Something More!” which might be a different title. So the next step is to search for the composer of ”Something Has To Separate” and see if that’s the same as the one for ”Something More!” or if there’s a different composer. 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: retrievesearch
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!’. The retrieval history shows that they’ve already done four steps of searching. The memory now states that Sammy Fain is the composer, with Robert Prince contributing as well.
...
The guidelines say to retrieve unless the memory clearly has all the info. The memory mentions Document 109 explicitly stating Sammy Fain as the composer. The other steps were about Syren’s album, 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 9: **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	59.72	58.99	**66.00**
HQA_28k	71.18	59.20	**71.44**
SQD_32k	**69.49**	61.70	65.31
MSQ_32k	41.79	46.27	**56.58**
2WK_32k	56.43	68.78	**70.66**
avg_LB	47.11	46.46	**50.93**
LB NQA	20.74	20.43	**20.77**
LB HQA	63.80	60.34	**65.14**
LB 2Wiki	67.83	65.19	**74.76**
LB Qasper	41.02	35.68	**40.74**
LB Musique	42.14	50.66	**53.22**

Table 8: **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 LongBench QA outcomes. Bold denotes the best score within this block, and underline denotes the runner-up.

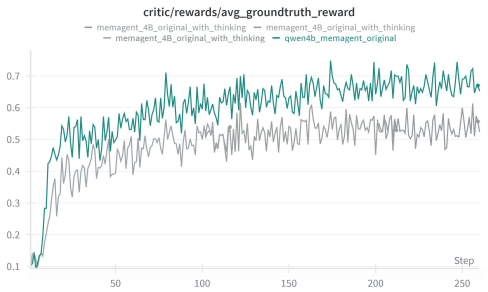

Figure 10: **Training dynamics: thinking vs. no-thinking for MemAgent-RL reproduction.** We plot the training curves of reproduced MemAgent-RL runs on Qwen3-4B with thinking mode enabled and disabled. During training, the no-thinking variant consistently outperforms the thinking variant.

MemAgent (Static Chunking: Redundant reasoning on irrelevant context)

<think>
Okay, let's tackle this problem. The question is asking which film Huh Jung directed that was released on August 17, 2017. .. Since there's no conflicting information in the provided section, the previous memory remains accurate. So, the answer should still be "The Mimic."
</think>

Key Details from the Section

- **Document 154** confirms *Hide and Seek* (2013) as Huh Jung's film, but this does not conflict with the previous memory.
- **No other films directed by Huh Jung in 2017** are mentioned in the provided section.
- **All other documents** in the section focus on unrelated topics, reaffirming the accuracy of the previous memory.

Updated Memory: Huh Jung directed "The Mimic," a horror film released on August 17, 2017.

InfMem (Dynamic Chunking: Targeted evidence composition and efficient memory update)

<think>
Okay, let's tackle this problem step by step. The question is asking which film directed by Huh Jung was released on August 17, 2017 .... The note in the memory also confirms that there's no conflicting info, so the conclusion is valid.
</think>

Key evidence from the retrieved chunks

- **Document 7 (retrieved chunk):** Confirms Huh Jung's second feature film is *The Mimic*, which is "scheduled to release in 2017." The memory explicitly states the release date as August 17, 2017.

Final Result

The film directed by Huh Jung released on August 17, 2017, is **"The Mimic"**.
Final note: The retrieved chunks confirm Huh Jung's film and its release date, supporting the existing memory.
Updated memory: Huh Jung directed the film *The Mimic*, which was released on August 17, 2017.

Figure 11: **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.

Query: Nick Garcia played soccer for which Major League Soccer team as a member of the Eastern Conference?

MemAgent (NoThink-RL) Process:
From Document 81, it is stated that Toronto FC is a team in the Major League Soccer Eastern Conference. However, this information is not directly applicable to Nick Garcia.
From Document 17, it is stated that the Columbus Crew is a team in the Major League Soccer Eastern Conference. However, this information is not directly applicable to Nick Garcia.
From Document 10, it is stated that Occhiuto played for the Buffalo Stallions ...
From Document 17, it is stated that the Columbus Crew is a team in the Major League Soccer Eastern Conference. [Repetition]
From Document 10, it is stated that Occhiuto played for the Buffalo Stallions of the Major Indoor Soccer League. However, this information is not relevant to the problem.
...(Repeats "Columbus Crew" and "Buffalo Stallions" which affect information density)

Query: Nick Garcia played soccer for which Major League Soccer team as a member of the Eastern Conference?

InfMem (Ours) Process:
<think>
Okay, let's tackle this problem step by step. ... I don't see any other mentions of Nick Garcia's team in the other documents. The other documents talk about other teams like Columbus Crew SC ... but not Nick Garcia. Therefore, the answer is Toronto FC.
</think>
The retrieved chunk (Document 109) explicitly states that **Nick Garcia played for Toronto FC**. Additionally, the memory section confirms that **Toronto FC is an MLS team in the Eastern Conference** (Document 81).
Updated Memory:
- Nick Garcia played for **Toronto FC** in MLS.
- **Toronto FC** is a member of the Eastern Conference.

Final Answer: Nick Garcia played soccer for **Toronto FC** in the Eastern Conference.

Figure 12: **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.