PI-MEM: PUSHING LONG-CONTEXT REASONING TO 3.6M TOKENS WITH PARALLEL-ITERATIVE MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context reasoning remains a critical bottleneck for large language models, as recent recurrent-memory approaches face two inherent challenges: sequential chunk-wise updates can overwrite early critical evidence with later irrelevant content, and serial inter-chunk dependencies limit parallelism and cause latency to increase with context length. To address these issues, we propose **PI-Mem** (Parallel-Iterative Memory), a mechanism that processes all chunks in parallel and iteratively refines a shared memory over a bounded number of turns. In each turn, PI-Mem reads all chunks in parallel conditioned on the current memory, selects new or complementary evidence from each chunk, and merges the selected evidence into a compact shared memory for the next turn. To discourage redundant turns, we optimize the workflow through reinforcement learning with an auxiliary turn-efficiency reward, enabling the model to adaptively exit once sufficient evidence has been accumulated. We evaluate PI-Mem with Qwen3.5-35B-A3B and Qwen2.5-7B on the HotpotQA benchmark across context lengths up to 3.6 million tokens and find that it outperforms the recurrent-memory baseline by +6.25 and +7.81 absolute points while achieving 6.1$\times$ and 2.1$\times$ inference speedups, respectively. These results demonstrate that PI-Mem breaks the accuracy–efficiency trade-off in long-context reasoning and provides a scalable approach to complex multi-hop question answering over extremely long documents.

1 INTRODUCTION

The ability to reason over long contexts is increasingly important for large language models (LLMs), supporting applications such as long-document question answering (Gemini Team, 2025; Yang et al., 2025b), multi-turn dialogue (Maharana et al., 2024; Li et al., 2024; Wu et al., 2025), repository-level code understanding (Yang et al., 2024; Hui et al., 2024), and agentic workflows over large external contexts (Yao et al., 2023; Agashe et al., 2024; Kimi Team et al., 2026). Yet model performance often degrades as context length grows, highlighting the persistent difficulty of identifying, retaining, and effectively utilizing relevant information across distant parts of the context (Hsieh et al., 2024; Du et al., 2025).

One family of approaches addresses this challenge by extending the natively supported context window through positional-encoding interpolation or extrapolation, thereby enabling direct processing of longer sequences without discarding input context (Chen et al., 2023; Peng et al., 2024; Liu et al., 2025a). In practice, such extensions are generally reliable only within a limited extrapolation range and still incur the high computational cost of dense attention at ultra-long context lengths. To address this efficiency bottleneck, sparse attention restricts computation to selected tokens or key-value blocks, whereas recurrent formulations of linear attention summarize preceding tokens in a compact state. Nevertheless, these architectures are costly to train and often rely on high-quality synthetic long-context corpora (Yuan et al., 2025; Lu et al., 2025; Yang et al., 2025c; Qwen Team, 2026), whose construction and curation remain difficult at scale across diverse reasoning tasks.

These limitations motivate recurrent-memory processing as a complementary strategy for handling long inputs. A recurrent-memory mechanism maintains a fixed-length textual memory as a compact state throughout input processing. At each step, the model processes the current chunk together with the previous memory and generates an updated memory that overwrites the previous state. Once all chunks have been processed, the model generates the final answer conditioned on the resulting mem-

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

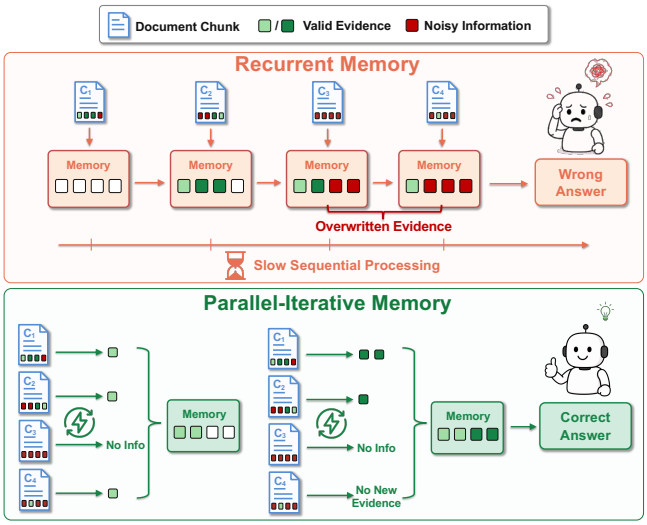


Figure 1: Top: Recurrent-memory workflows process chunks sequentially and repeatedly rewrite a running memory state, potentially overwriting early evidence and creating strict serial dependencies. Bottom: PI-Mem reads chunks in parallel conditioned on a shared global memory, improving evidence preservation and reducing inference latency.

ory (Yu et al., 2025a; Sheng et al., 2026; Shi et al., 2025). In principle, bounding the context of each memory update allows the workflow to process inputs of arbitrary length at a computational cost that grows linearly rather than quadratically with input length. However, as illustrated in Figure 1, this recurrent-memory design faces two challenges. First, successive memory updates compress each new chunk and the existing fixed-length memory into a new state, potentially overwriting early evidence with later noisy information before the relevance of the early evidence becomes apparent and thereby impairing evidence preservation and integration across distant chunks. Second, the recurrent-memory workflow imposes a strict inter-chunk dependency, forcing each chunk to await the preceding memory update and increasing inference latency with context length.

To address these challenges, we introduce PI-Mem, a Parallel-Iterative Memory mechanism for long-context reasoning. PI-Mem gathers evidence from all chunks in *parallel* and uses it to update a shared memory over a bounded number of turns. Specifically, we define each memory-update *turn* as a complete *read-select-merge* cycle. The read step processes all chunks in parallel, with each chunk-level read conditioned on the same global memory, thereby avoiding sequential memory overwrites that may discard earlier evidence and reducing inference latency through parallel chunk processing. PI-Mem then selects observations that provide new or complementary evidence and merges them with the current memory to produce an updated global memory. In the next turn, all chunks are read again conditioned on the updated memory, allowing evidence discovered in one chunk to guide extraction from other chunks and thereby enabling cross-chunk information exchange. This cycle repeats until either no chunk yields useful new evidence or a predefined maximum number of turns is reached, after which PI-Mem generates the final answer from the question and the consolidated global memory. We use end-to-end reinforcement learning (RL) to reward accuracy and fewer turns.

Experiments with Qwen3.5-35B-A3B and Qwen2.5-7B demonstrate the effectiveness and efficiency of PI-Mem. On HotpotQA (HQA) at 3.6M tokens, PI-Mem improves over MemAgent by +6.25 and +7.81 absolute points and delivers inference speedups of 6.1× and 2.1×, respectively.

Our main contributions are summarized as follows:

- We propose PI-Mem, a parallel-iterative memory mechanism for long-context reasoning that reads all chunks in parallel conditioned on a shared memory and iteratively refines the memory over a bounded number of turns.
- We optimize the workflow end-to-end with RL to improve long-context performance and introduce a turn-efficiency reward that discourages redundant turns.

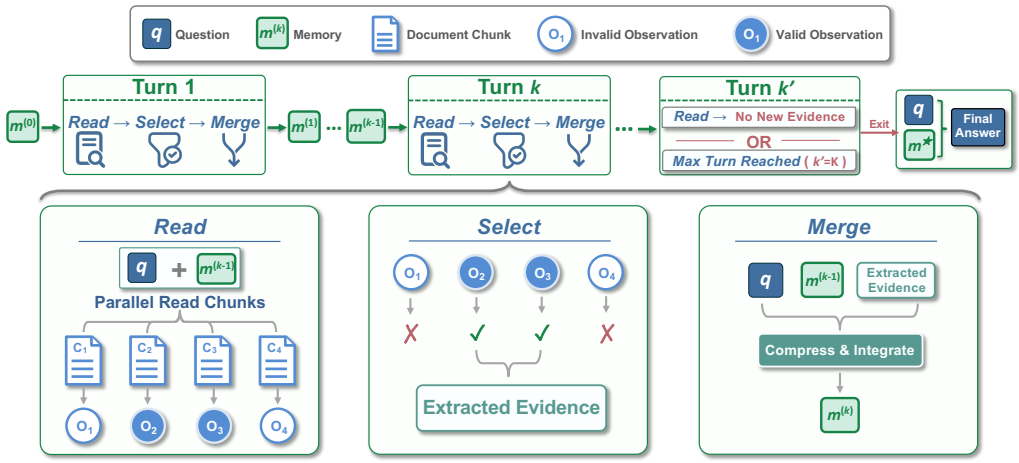


Figure 2: Overview of the PI-Mem workflow. PI-Mem reads all chunks in parallel conditioned on a shared global memory and iteratively refines this memory through read-select-merge turns.

- We empirically show that PI-Mem outperforms the recurrent-memory baseline and substantially reduces inference latency on HotpotQA at context lengths up to 3.6M tokens, while also improving performance across diverse long-context benchmarks.

2 Related Work

Long-Context Processing Frameworks. Memory-based systems extend context with external stores or compact recurrent states (Packer et al., 2023; Chhikara et al., 2025; Yu et al., 2025a; Zhou et al., 2025b; Chen et al., 2026b). GRU-Mem gates memory updates, whereas MHM-LRU protects multiple memory heads from overwrite (Sheng et al., 2026; Li et al., 2026). LycheeMemory compresses chunk KV caches in parallel but then scans selected blocks with sequential working-memory updates (Chen et al., 2026c). Other work improves memory construction, retrieval, and routing (Wang et al., 2025b; Zhang et al., 2026b; Wang et al., 2026; Sun et al., 2025; Ye et al., 2025; Feng et al., 2026). Other long-text workflows use leader-coordinated member agents across multiple rounds or map–reduce aggregation of chunk outputs (Zhao et al., 2024; Zhou et al., 2025a); recursive models invoke calls over selected portions of the input (Zhang et al., 2026a). Parallel query-guided segment compression retains relevant text while compressing less relevant segments into summary vectors before final decoding (Tang et al., 2026). PI-Mem differs in how it shares information across chunks: the same explicit shared memory conditions every parallel chunk read in each turn. After selected observations are merged, the updated memory conditions a full reread of all chunks in the next turn.

Reinforcement Learning for Long-Context Reasoning. RL with verifiable rewards, together with group-relative policy optimization and its variants, has substantially improved multi-step reasoning (Shao et al., 2024; Yu et al., 2025b). Long-context studies use SFT warm-up, curriculum-guided RL, difficulty-aware sampling, self-play question generation and verification, and synthetic multi-hop QA data (Wan et al., 2025; Yang et al., 2026; Wang et al., 2025a). Beyond outcome-only supervision, dense, verifiable context rewards directly guide grounding and evidence selection (Chen et al., 2026a). RL also supports multi-stage memory fusion and trains agents to revisit earlier memories or control when to retrieve, write, and stop (Shen et al., 2025; Shi et al., 2025; Wang et al., 2026). Our approach applies trajectory-level RL to a parallel-iterative workflow, training the policy to coordinate evidence selection and memory refinement and discouraging redundant turns.

3 Method

In this section, we first describe the PI-Mem inference workflow and then present its RL training procedure.

Algorithm 1 PI-Mem Inference Workflow

Input: Question q, context C, model π_θ, maximum number of turns K
Output: Final answer a
Initialize: Split C into chunks $\{c_i\}_{i=1}^n$, $m^{(0)} \leftarrow$ `EMPTY`

$m^\star \leftarrow m^{(0)}$
for $k = 1$ to K **do**
  Read each chunk c_i in *parallel*: $o_i^{(k)} \leftarrow \textsc{ReadCall}(\pi_\theta, q, c_i, m^{(k-1)})$
  $O^{(k)} \leftarrow \{\, o_i^{(k)} \mid \chi(o_i^{(k)}) = \texttt{yes} \,\}$
  if $O^{(k)} = \emptyset$ **then**
    break
  else
    $m^{(k)} \leftarrow \textsc{MergeCall}(\pi_\theta, q, m^{(k-1)}, O^{(k)})$
    $m^\star \leftarrow m^{(k)}$
  end if
end for
$a \leftarrow \textsc{FinalCall}(\pi_\theta, q, m^\star)$
return a

3.1 PI-Mem Inference Workflow

Figure 2 provides an overview of the PI-Mem inference workflow, and Algorithm 1 presents its procedural details. PI-Mem reads all chunks in parallel and iteratively refines a shared memory over a bounded number of turns. We define each memory-update *turn* as a complete *read-select-merge* cycle: PI-Mem reads all chunks conditioned on the current global memory, selects observations with new or complementary evidence, and merges them into an updated global memory. We first detail the three operations within each turn and then describe iterative refinement and adaptive exit across turns.

Read-Select-Merge Turn. For a general QA task, we denote the question by q, the full background context by C, and the model by π_θ. We first split C into equal-sized chunks $\{c_i\}_{i=1}^n$ and initialize the global memory $m^{(0)}$ as an empty memory. At turn k, PI-Mem updates the global memory through a *read-select-merge* turn:

$$m^{(k)} = \textsc{Turn}(\pi_\theta, q, m^{(k-1)}, \{c_i\}_{i=1}^n). \tag{1}$$

The goal of each turn is to update the global memory by processing all chunks in parallel.

In the **read** step, PI-Mem processes all chunks in parallel. For each chunk c_i, ReadCall places q, c_i, and $m^{(k-1)}$ into a chunk-reading template and calls π_θ to generate a chunk-level observation $o_i^{(k)}$. Rather than successively overwriting the memory as in sequential recurrent processing, PI-Mem reads all chunks independently conditioned on the same memory $m^{(k-1)}$, thereby reducing the risk that evidence from early chunks is overwritten by noisy information from later chunks. This independence also enables PI-Mem to read chunks in parallel and efficiently batch the corresponding calls, allowing ultra-long sequences to be processed with lower inference latency. Each chunk call must explicitly output a check signal, either `<check>yes</check>` or `<check>no</check>`. The model outputs `yes` when c_i provides evidence relevant to q that is new or complementary to $m^{(k-1)}$. When the check signal is `yes`, the output includes the corresponding evidence needed for answering q; otherwise, the model outputs only `no`.

The **select** step filters observations by the `<check>` signal. Let $\chi(o_i^{(k)})$ denote the value enclosed by the check tags; the selected observation set is defined as

$$O^{(k)} = \{\, o_i^{(k)} \mid \chi(o_i^{(k)}) = \texttt{yes} \,\}. \tag{2}$$

This filtering removes observations that provide no useful update, so the memory update focuses on new or complementary evidence rather than irrelevant or redundant text.

The **merge** step consolidates the selected observations into the updated global memory. Specifically, MergeCall places q, $m^{(k-1)}$, and $O^{(k)}$ into a merge template and invokes π_θ to produce $m^{(k)}$.

A straightforward alternative is to concatenate all positive chunk outputs, but this can still produce an overly long intermediate context when many chunks contain relevant evidence. The merge step instead compresses and integrates evidence from different chunks, removes redundancy, preserves details necessary for answering q, and prevents unbounded memory growth as the number of selected observations increases. Thus, a single turn updates the global memory through parallel evidence extraction, selective filtering, and compact cross-source integration.

Iterative Refinement and Exit Mechanism. After turn k finishes, PI-Mem feeds the merged memory $m^{(k)}$ into the next read step and repeats the read-select-merge cycle. Each later turn therefore reads all chunks again, conditioned on evidence discovered in previous turns. Compared with a single greedy turn, in which chunk relevance is judged only from the initial empty memory, iterative refinement reduces the risk of overlooking evidence whose importance becomes clear only after other facts have been identified and enables cross-chunk information exchange through the shared memory.

PI-Mem further uses the check signal as an **adaptive exit mechanism**. The loop automatically stops when no chunk produces useful new evidence, i.e., $O^{(k)} = \emptyset$, or when it reaches the maximum number of turns K. Finally, PI-Mem invokes the model once more to answer using only the question q and the final global memory $m^\star$, rather than the original long context, so that final inference remains compact and grounded in the consolidated evidence.

3.2 Trajectory-Level Reinforcement Learning

We adopt GRPO (Shao et al., 2024) to optimize the multi-call workflow end to end. During RL training, each rollout for a query defines a complete trajectory. Since the workflow invokes the model multiple times, each trajectory contains multiple model-call samples, including read, merge, and final-answer calls. We denote the i-th trajectory by

$$\tau_i = \{(x_{i,s}, y_{i,s})\}_{s=1}^{S_i}, \tag{3}$$

where S_i is the number of model-call samples in trajectory τ_i, s indexes one such model call, $x_{i,s}$ is the corresponding phase-specific prompt constructed from the workflow state, and $y_{i,s}$ is the generated response. For each training query, we sample a group of G complete trajectories $\{\tau_i\}_{i=1}^{G}$ from the old policy $\pi_{\theta_{\text{old}}}$ and use all model-call samples in these trajectories for policy optimization. We next describe how the trajectory-level reward is constructed and used for end-to-end workflow optimization.

Reward Design. We assign one reward to each completed trajectory τ_i based on its final answer a_i and turn count k_i. Let $r_{\text{acc}}(a_i)$ denote the accuracy reward and K the maximum number of turns. We augment answer accuracy with a turn-efficiency bonus:

$$r_{\text{turn}}(\tau_i) = \frac{K - k_i}{K - 1};\; R_i = r_{\text{acc}}(a_i) + \lambda_{\text{turn}} r_{\text{turn}}(\tau_i). \tag{4}$$

The turn bonus assigns higher rewards to trajectories that exit in fewer turns, and λ_{turn} controls its contribution. It discourages redundant refinement once the memory is sufficient and encourages the workflow to exit when no new evidence is found, reducing latency.

Trajectory-Level Optimization. This scalar reward is then broadcast to every model-call sample $(x_{i,s}, y_{i,s})$ in the same trajectory. Following Dr. GRPO (Liu et al., 2025b), we omit normalization by the group standard deviation and compute the group-relative advantage over complete trajectories rather than individual calls:

$$A_i = R_i - \frac{1}{G}\sum_{j=1}^{G} R_j. \tag{5}$$

Given this trajectory-level advantage, each token in every model-call sample of trajectory τ_i uses the same A_i in the per-token clipped surrogate term:

$$\ell_{i,s,t}(\theta) = \min\Big(\rho_{i,s,t}(\theta)A_i, \text{clip}(\rho_{i,s,t}(\theta), 1-\varepsilon, 1+\varepsilon)A_i\Big), \tag{6}$$

where ε denotes the clipping ratio, and the token-level policy ratio is

$$\rho_{i,s,t}(\theta) = \frac{\pi_\theta(y_{i,s,t} \mid x_{i,s}, y_{i,s,<t})}{\pi_{\theta_{\text{old}}}(y_{i,s,t} \mid x_{i,s}, y_{i,s,<t})}. \tag{7}$$

The final objective uses DAPO-style token-level aggregation Yu et al. (2025b) over all model-call samples in the sampled trajectories. Let $Z = \sum_{i=1}^{G} \sum_{s=1}^{S_i} |y_{i,s}|$ denote the total number of generated tokens. The objective is

$$\mathcal{J}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{y_{i,s}\}\sim\pi_{\theta_{\text{old}}}(\cdot|x_{i,s})} \left[\frac{1}{Z} \sum_{i=1}^{G} \sum_{s=1}^{S_i} \sum_{t=1}^{|y_{i,s}|} \Big(\ell_{i,s,t}(\theta) - \beta \mathrm{KL}(\pi_\theta \| \pi_{\text{ref}}) \Big) \right]. \tag{8}$$

Here, (q, a) denotes a question and its reference answer sampled from the data distribution $\mathcal{D}$. The coefficient β scales the token-level KL divergence between the current policy π_θ and the reference policy π_{ref}.

4 Experiments

4.1 Training Setup

We evaluate our method on two representative models: Qwen3.5-35B-A3B (Qwen Team, 2026), an MoE model with hybrid attention, and Qwen2.5-7B-Instruct (Yang et al., 2025a), a dense model with full attention.

For Qwen3.5-35B-A3B, we follow MemAgent's data construction (Yu et al., 2025a), embedding gold HotpotQA (Yang et al., 2018) paragraphs among distractor documents. Each sample contains 1,000 documents (approximately 140K tokens). We use 15K-token chunks, a 4,096-token output limit at every generation stage, $K = 3$ turns, and disable thinking during training. MemAgent is trained on the same data with identical RL hyperparameters for a fair comparison. Full settings, including rollout sizes and steps, are provided in Appendix A.

For Qwen2.5-7B, each HotpotQA training sample contains 200 documents (approximately 28K tokens). PI-Mem uses 5K-token chunks and a maximum output length of 1,024 tokens. We use 16 rollouts per prompt and 240 rollout steps, and compare against the officially released MemAgent checkpoint with the same backbone.

4.2 Evaluation Setup

Baselines. We compare PI-Mem with direct-inference, retrieval-augmented, and recurrent-memory baselines. Vanilla denotes standard direct inference using the backbone model. We also include YaRN (Peng et al., 2024), a commonly used positional-encoding extrapolation method, as a direct-inference baseline. RAG (Lewis et al., 2020) retrieves relevant chunks from the long context and performs direct inference over the retrieved evidence. We include MemAgent (Yu et al., 2025a), GRU-Mem (Sheng et al., 2026), ReMemR1 (Shi et al., 2025), and LycheeMemory (Chen et al., 2026c) as memory-based baselines. We report the **RL-trained variants** of MemAgent and PI-Mem and use the published HQA results for GRU-Mem, ReMemR1, and LycheeMemory. For LycheeMemory, we use its highest-scoring variant without the Gate, labeled Lychee in Table 1.

Benchmarks. We evaluate long-context performance on RULER (Hsieh et al., 2024) and LongBench v2 (Bai et al., 2024). RULER is a synthetic benchmark that probes retrieval, multi-hop tracing, aggregation, and question answering over long contexts. LongBench v2 contains 503 challenging multiple-choice questions across six realistic task categories. In our RULER results, all tasks *except HQA* are treated as out-of-distribution (OOD) tasks, while HQA is considered in-distribution because the RL training data is synthesized from HQA. We report LongBench v2 results only for Qwen3.5, as all evaluated methods on Qwen2.5 performed at approximately the random-guessing level on this four-choice benchmark in our experiments, making method-level comparisons difficult to interpret.

Table 1: Length-grouped RULER HQA results. Scores are reported across context lengths from 7K to 3.6M tokens. Asterisks (*) mark results from the respective original papers.

Method	7K	14K	28K	56K	112K	224K	448K	896K	1.8M	3.6M	Avg.
Qwen3.5-35B-A3B											
Vanilla	**87.50**	79.69	82.81	81.25	78.12	73.44	62.50	37.50	34.38	29.69	64.69
YaRN	82.81	81.25	79.69	**82.81**	78.12	68.75	59.38	56.25	29.69	14.06	63.28
RAG	84.38	**82.81**	71.88	67.19	46.88	54.69	51.56	53.12	46.88	43.75	60.31
MemAgent	84.38	81.25	82.81	81.25	79.69	68.75	73.44	71.88	71.88	70.31	76.56
PI-Mem	**87.50**	**82.81**	**84.38**	**82.81**	**84.38**	**79.69**	**82.81**	**76.56**	**75.00**	**76.56**	**81.25**
Qwen2.5-7B											
Vanilla	60.94	54.69	50.00	23.44	0.00	0.00	0.00	0.00	0.00	0.00	18.91
YaRN	56.25	62.50	62.50	45.31	34.38	0.00	0.00	0.00	0.00	0.00	26.09
RAG	57.81	46.88	46.88	45.31	39.06	42.19	29.69	43.75	26.56	32.81	41.09
MemAgent	81.25	81.25	75.00	**82.81**	76.56	75.00	76.56	75.00	78.12	73.44	77.50
Lychee*	80.47	81.25	**82.03**	81.25	80.47	79.69	75.00	75.78	78.12	N/A	79.34
GRU-Mem*	81.25	79.69	77.34	76.56	74.22	73.44	71.88	76.56	N/A	N/A	76.37
ReMemR1*	82.3	82.8	81.1	78.9	82.0	79.7	80.0	80.8	N/A	N/A	80.95
PI-Mem	**82.81**	**82.81**	81.25	**82.81**	**89.06**	**85.94**	**82.81**	**84.38**	**87.50**	**81.25**	**84.06**

Implementation Details. Thinking mode is disabled for Qwen3.5 and is not applicable to Qwen2.5. We use temperature $= 0.7$ and top-$p = 0.95$, and set the maximum output length to 4,096 tokens for Qwen3.5 and 1,024 tokens for Qwen2.5. We use 8 NVIDIA H200 GPUs with tensor parallelism set to 2 for inference. For RULER, each task at each context length is evaluated with 64 samples.

4.3 Main Results

Tables 1 and 2 show that PI-Mem achieves the best HQA average for both backbones, with gains over MemAgent especially at long contexts. It remains effective at 3.6M tokens, where Vanilla and YaRN degrade sharply. On RULER OOD, PI-Mem also leads on average and maintains high 1M-token performance for both backbones. These gains beyond HQA-based training indicate robustness to context length and transfer to diverse tasks.

Table 3 reports the LongBench v2 results. PI-Mem achieves the best overall score of 54.1, outperforming MemAgent by 1.8 points and the vanilla baseline by 3.4 points.

Evidence Retention Analysis. To assess whether recurrent updates overwrite early evidence, we analyze RULER MV-NIAH (Hsieh et al., 2024) with Qwen3.5. Each example scatters four values associated with the same key across widely separated positions in a long context; counting how many remain in the final memory after all chunks have been processed provides a direct measure of evidence retention. To probe parallel reading without PI-Mem's selection or iteration, we include a parallel-only variant. It reads all chunks once in parallel, without a check signal or further turns, then merges their outputs into final memory. Table 4 shows that both parallel-only and PI-Mem retain nearly all values, whereas recurrent MemAgent retains substantially fewer. This comparison indicates that parallel reading preserves the collected evidence more effectively by mitigating the overwrite caused by recurrent memory updates.

Latency Analysis. Figure 3 compares end-to-end inference latency on ultra-long HQA inputs. At 3.6M tokens, PI-Mem is 16.9× faster than Vanilla and 6.1× faster than MemAgent with Qwen3.5-35B-A3B; the corresponding speedups with Qwen2.5-7B are 14.4× and 2.1×. Vanilla keeps the full context active during inference, incurring substantial full-context prefill and KV-cache costs. MemAgent bounds the context of each call through chunking, but its recurrent updates form a serial critical path that prevents concurrent processing of chunks from the same sequence. PI-Mem removes this dependency by executing chunk-level ReadCall operations in parallel within each turn and batching the corresponding calls across examples, thereby improving GPU utilization and reducing latency.

Table 2: Length-grouped RULER out-of-distribution results. Scores are reported across context lengths from 8K to 1M tokens.

Method	8K	16K	32K	64K	128K	256K	512K	1M	Avg.
Qwen3.5-35B-A3B									
Vanilla	98.67	98.39	97.64	97.96	97.40	96.08	95.81	75.56	94.69
YaRN	97.92	96.59	96.74	97.02	97.46	97.22	96.19	86.58	95.72
RAG	98.31	93.07	94.32	95.18	93.25	87.02	77.90	75.24	89.29
MemAgent	94.55	92.92	95.18	94.05	92.46	90.84	86.42	83.86	91.28
PI-Mem	**98.92**	**98.65**	**98.06**	**98.42**	**97.69**	**98.38**	**97.30**	**96.88**	**98.04**
Qwen2.5-7B									
Vanilla	86.57	87.17	84.48	60.65	22.72	0.00	0.00	0.00	42.70
YaRN	84.24	83.03	78.82	72.48	57.13	0.00	0.00	0.00	46.96
RAG	80.58	79.30	77.57	76.49	74.07	68.05	63.45	58.70	72.28
MemAgent	91.66	90.33	89.31	87.73	83.73	80.30	79.51	74.83	84.68
PI-Mem	**91.72**	**91.78**	**92.10**	**92.49**	**91.75**	**91.38**	**89.14**	**88.39**	**91.09**

Table 3: LongBench v2 results. Following the grouping defined by the benchmark, we report results by difficulty (Easy and Hard) and context length (Short, Medium, and Long).

Method	**All**	**Difficulty**		**Length**		
		Easy	Hard	Short	Med.	Long
Vanilla	50.7	55.7	47.6	56.1	47.9	47.2
YaRN	51.5	52.1	**51.1**	**57.8**	48.4	47.2
RAG	48.9	53.6	46.0	51.7	44.7	52.8
MemAgent	52.3	59.9	47.6	55.6	48.8	53.7
PI-Mem	**54.1**	**66.1**	46.6	53.3	**49.8**	**63.9**

Table 4: Final-memory evidence coverage on MV-NIAH, measured as the proportion of ground-truth values present in the final memory.

Method	256K	512K	1M
MemAgent	70.3%	74.2%	58.6%
Parallel-only	100%	99.2%	100%
PI-Mem	100%	100%	100%

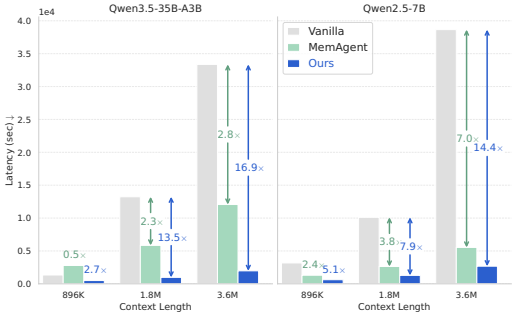


Figure 3: End-to-end HQA inference latency (sec) at ultra-long context lengths with Qwen3.5-35B-A3B and Qwen2.5-7B. Arrows show speedups relative to Vanilla. At 3.6M tokens, PI-Mem's speedups over MemAgent are 6.1× for Qwen3.5 and 2.1× for Qwen2.5. PI-Mem reduces latency by parallelizing chunk-level reads instead of serially updating memory across chunks.

4.4 Ablation Study

For computational efficiency, we conduct all ablations in this section with Qwen2.5-7B-Instruct.

Effect of RL Training. Figure 4 evaluates whether RL training improves the proposed workflow on RULER OOD tasks. RL training yields only a minor gain at 8K tokens, but the improvement becomes increasingly clear as the context length grows. This trend suggests that even with simple synthetic data based on HQA, RL training helps the model generalize to more diverse long-context tasks and use the workflow more effectively.

Select and Merge Components. Figure 5 ablates selection and merging on HQA at 896K, 1.8M, and 3.6M tokens. Without select, no <check> signal is required and all chunk observations reach the merge step; without merge, selected observations are concatenated rather than consolidated. Removing either component weakens context management: without selection, irrelevant, redundant, or noisy observations accumulate, whereas without merging, useful evidence remains fragmented and memory expands rapidly. As the input grows, this context pollution or fragmentation leads to lower accuracy and longer inference time.

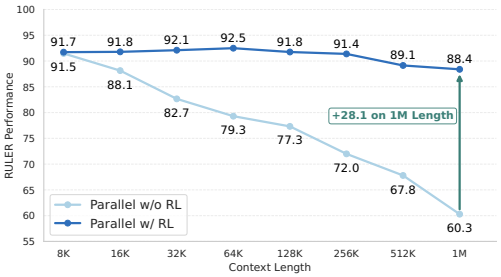


Figure 4: Effect of RL training on RULER OOD performance across context lengths. RL training brings larger gains as the context becomes longer.

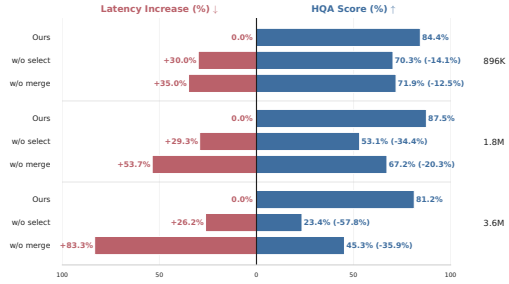


Figure 5: Component ablation on HQA. HQA Score reports the absolute score, whereas Latency Increase reports the relative percentage increase over the full configuration.

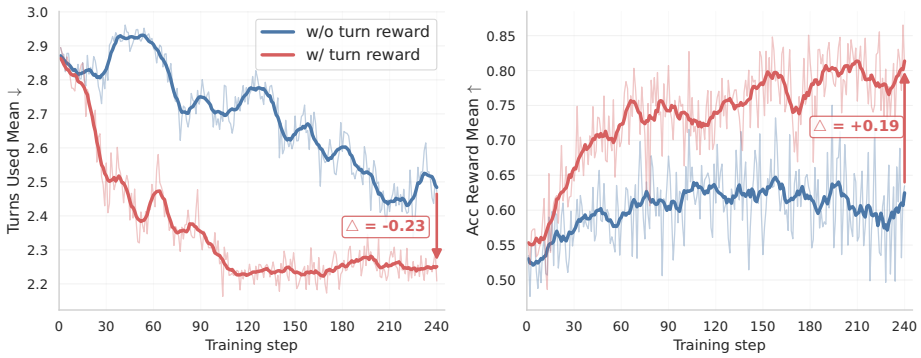


Figure 6: Effect of the turn-efficiency reward on Qwen2.5-7B RL training. The turn reward leads to a faster, larger, and more stable reduction in the number of turns, while also improving the accuracy reward.

Turn-Efficiency Reward. Figure 6 compares training with and without the turn-efficiency reward ($\lambda_{\text{turn}} = 0.2$ when enabled). Both variants learn to use fewer read-select-merge turns, but without the reward the decline is smaller and less stable. The reward quickly suppresses redundant turns and converges to a lower, more stable count (left). It also raises the late-stage accuracy reward (right), so the efficiency gain does not sacrifice correctness. Before workflow-level training, extra turns can be unnecessary or harmful because repeated updates may introduce noise, overwrite useful evidence, or over-refine sufficient memory.

Further analyses in the Appendix include more implementation details, turn-budget (K) ablations, actual turn counts, detailed comparisons of inference FLOPs and end-to-end latency, and comparative case studies.

5 Conclusion

We introduced PI-Mem, a parallel-iterative memory mechanism for long-context reasoning that gathers evidence from all chunks in parallel and iteratively refines a shared memory. Instead of updating memory recurrently after every chunk, PI-Mem reads chunks in parallel conditioned on a shared global memory, selects useful observations, and merges them into a compact memory across turns. We further post-trained the workflow with RL so the model can better follow the read-select-merge process and avoid unnecessary turns. Experiments with Qwen3.5-35B-A3B and Qwen2.5-7B demonstrate that PI-Mem improves long-context QA and retrieval performance while reducing inference latency compared with recurrent-memory and direct-inference baselines. Together, memory workflow design and end-to-end post-training offer a promising path to ultra-long-context reasoning without relying solely on larger native context windows.

AI USE STATEMENT

LLMs were used to assist with writing and polishing the paper. We used them to improve wording, readability, and clarity throughout the manuscript. The authors independently developed all research concepts and ideas and conducted all analyses. The authors take full responsibility for the paper's content, including any text generated or polished with LLM assistance. We reviewed the LLM-assisted text to ensure that it meets ethical standards and contains no plagiarized material or fabricated claims. LLMs also assisted in organizing experimental results and populating LaTeX tables; all reported numerical values came from actual experiments and were not generated or fabricated by LLMs.

ETHICS STATEMENT

This work uses existing, publicly available QA benchmarks and involves no human-subject study. Long-context answers may still be inaccurate or reflect biases in the underlying models and data; our results do not establish suitability for high-stakes decisions.

REPRODUCIBILITY STATEMENT

The main text specifies the PI-Mem workflow and experimental protocol. Appendix A reports training data sources and hyperparameters, while Appendix B details evaluation settings and metrics. Prompt templates and additional results are also included in the Appendix.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human. *arXiv preprint arXiv:2410.08164*, 2024.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*, 2024.

Guanzheng Chen, Michael Qizhe Shieh, and Lidong Bing. Longrlvr: Long-context reinforcement learning requires verifiable context rewards. *arXiv preprint arXiv:2603.02146*, 2026a.

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Wayne Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, Kuan Li, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Iterresearch: Rethinking long-horizon agents with interaction scaling, 2026b. URL https://arxiv.org/abs/2511.07327.

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

Zhuoen Chen, Dongfang Li, Meishan Zhang, Baotian Hu, and Min Zhang. Dynamic long context reasoning over compressed memory via end-to-end reinforcement learning, 2026c. URL https://arxiv.org/abs/2602.08382.

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

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. Context length alone hurts llm performance despite perfect retrieval, 2025. URL https://arxiv.org/abs/2510.05381.

Zhaopeng Feng, Liangcai Su, Zhen Zhang, Xinyu Wang, Xiaotian Zhang, Xiaobin Wang, Runnan Fang, Qi Zhang, Baixuan Li, Shihao Cai, Rui Ye, Hui Chen, Jiang Yong, Joey Tianyi Zhou, Chenxiong Qian, Pengjun Xie, Bryan Hooi, Zuozhu Liu, and Jingren Zhou. Agentswing: Adaptive parallel context management routing for long-horizon web agents, 2026. URL https://arxiv.org/abs/2603.27490.

Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models?, 2024. URL https://arxiv.org/abs/2404.06654.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, et al. Kimi k2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

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

Hao Li, Chenghao Yang, An Zhang, Yang Deng, Xiang Wang, and Tat-Seng Chua. Hello again! llm-powered personalized agent for long-term dialogue. *arXiv preprint arXiv:2406.05925*, 2024.

Jiatong Li, Samuel Yeh, and Sharon Li. Multi-head recurrent memory agents, 2026. URL https://arxiv.org/abs/2607.01523.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, Yuanxing Zhang, Zhuo Chen, Hangyu Guo, Shilong Li, Ziqiang Liu, Yong Shan, Yifan Song, Jiayi Tian, Wenhao Wu, Zhejian Zhou, Ruijie Zhu, Junlan Feng, Yang Gao, Shizhu He, Zhoujun Li, Tianyu Liu, Fanyu Meng, Wenbo Su, Yingshui Tan, Zili Wang, Jian Yang, Wei Ye, Bo Zheng, Wangchunshu Zhou, Wenhao Huang, Sujian Li, and Zhaoxiang Zhang. A comprehensive survey on long context language modeling, 2025a. URL https://arxiv.org/abs/2503.17407.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025b.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

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

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

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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

Weizhou Shen, Ziyi Yang, Chenliang Li, Zhiyuan Lu, Miao Peng, Huashan Sun, Yingcheng Shi, Shengyi Liao, Shaopeng Lai, Bo Zhang, Dayiheng Liu, Fei Huang, Jingren Zhou, and Ming Yan. Qwenlong-l1.5: Post-training recipe for long-context reasoning and memory management. *arXiv preprint arXiv:2512.12967*, 2025.

Leheng Sheng, Yongtao Zhang, Wenchang Ma, Yaorui Shi, Ting Huang, Xiang Wang, An Zhang, Ke Shen, and Tat-Seng Chua. When to memorize and when to stop: Gated recurrent memory for long-context reasoning, 2026. URL https://arxiv.org/abs/2602.10560.

Yaorui Shi, Yuxin Chen, Siyuan Wang, Sihang Li, Hengxing Cai, Qi Gu, Xiang Wang, and An Zhang. Look back to reason forward: Revisitable memory for long-context llm agents. *arXiv preprint arXiv:2509.23040*, 2025.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025.

Jiwei Tang, Shilei Liu, Zhicheng Zhang, Qingsong Lv, Runsong Zhao, Tingwei Lu, Langming Liu, Haibin Chen, Yujin Yuan, Hai-Tao Zheng, Wenbo Su, and Bo Zheng. Read as human: Compressing context via parallelizable close reading and skimming, 2026. URL https://arxiv.org/abs/2602.01840.

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, 2025. URL https://arxiv.org/abs/2505.17667.

Siyuan Wang, Gaokai Zhang, Li Lyna Zhang, Ning Shang, Fan Yang, Dongyao Chen, and Mao Yang. Loongrl: Reinforcement learning for advanced reasoning over long contexts. *arXiv preprint arXiv:2510.19363*, 2025a.

Xinyu Wang, Mingze Li, Peng Lu, Xiao-Wen Chang, Lifeng Shang, Jinping Li, Fei Mi, Prasanna Parthasarathi, and Yufei Cui. Infmem: Learning system-2 memory control for long-context agent. *arXiv preprint arXiv:2602.02704*, 2026.

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

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory, 2025. URL https://arxiv.org/abs/2410.10813.

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, Tianyi Tang, 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, 2025a. URL https://arxiv.org/abs/2412.15115.

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

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025c. URL https://arxiv.org/abs/2412.06464.

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, 2018. URL https://arxiv.org/abs/1809.09600.

Ziyi Yang, Weizhou Shen, Chenliang Li, Ruijun Chen, Fanqi Wan, Ming Yan, Xiaojun Quan, and Fei Huang. Spell: Self-play reinforcement learning for evolving long-context language models, 2026. URL https://arxiv.org/abs/2509.23863.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, Pengjun Xie, Fei Huang, Siheng Chen, Jingren Zhou, and Yong Jiang. Agentfold: Long-horizon web agents with proactive context management, 2025. URL https://arxiv.org/abs/2510.24699.

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. *arXiv preprint arXiv:2507.02259*, 2025a.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025b. URL https://arxiv.org/abs/2503.14476.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

Alex L. Zhang, Tim Kraska, and Omar Khattab. Recursive language models, 2026a. URL https://arxiv.org/abs/2512.24601.

Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks, 2026b. URL https://arxiv.org/abs/2510.12635.

Jun Zhao, Can Zu, Xu Hao, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. LONGAGENT: Achieving question answering for 128k-token-long documents through multi-agent collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16310–16324, 2024. doi: 10.18653/v1/2024.emnlp-main.912. URL https://aclanthology.org/2024.emnlp-main.912/.

Zihan Zhou, Chong Li, Xinyi Chen, Shuo Wang, Yu Chao, Zhili Li, Haoyu Wang, Qi Shi, Zhixing Tan, Xu Han, Xiaodong Shi, Zhiyuan Liu, and Maosong Sun. LLM×MapReduce: Simplified long-sequence processing using large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27664–27678, 2025a. doi: 10.18653/v1/2025.acl-long.1341. URL https://aclanthology.org/2025.acl-long.1341/.

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*, 2025b.

A Training Details

Table 5: Training hyperparameters for PI-Mem with Qwen3.5-35B-A3B and Qwen2.5-7B-Instruct.

Hyperparameter	**Qwen3.5**	**Qwen2.5**
Documents per sample	1,000	200
Context length	140K	28K
Chunk size	15,000	5,000
Maximum output length	4,096	1,024
Maximum turns (K)	3	
Train batch size	128	
Mini batch size	8	
Adam optimizer β_1, β_2	$(0.9,\ 0.999)$	
Adam optimizer ϵ	1×10^{-8}	
Weight decay	0.01	
Gradient clipping	1.0	
Warmup steps	20	
Learning rate	1×10^{-6}	
Learning-rate schedule	Constant after linear warmup	
KL loss coefficient (β)	1×10^{-3}	
Clip ratio (ε)	0.2	
MoE router auxiliary loss coef.	1×10^{-3}	N/A
GRPO group size (G)	8	16
Training rollout steps	80	240
Training samples	10,240	30,720
Temperature	1.0	
Top-p	1.0	
Thinking mode	Disabled	N/A
GPU world size (H200)	32	16
Training Time (hours)	75	96

Qwen2.5-7B uses MemAgent's released training data; for Qwen3.5-35B-A3B's longer native context, we synthesize longer HQA samples with the same procedure. Most RL hyperparameters and algorithmic choices follow MemAgent, including Dr. GRPO-style advantage normalization (without division by the group standard deviation) and DAPO-style loss aggregation. We reduce the Qwen3.5 GRPO group size because of its higher training cost, and train PI-Mem and MemAgent on identical Qwen3.5 data and hyperparameters.

B Evaluation Details

B.1 Evaluation Setup

Table 6: Evaluation configurations for PI-Mem with Qwen3.5-35B-A3B and Qwen2.5-7B-Instruct.

Configuration	**Qwen3.5**	**Qwen2.5**
Chunk size	15,000	5,000
Maximum output length	4,096	1,024
Maximum turns (K)	3	
GPU world size (H200)	8	
Tensor parallel size	2	
Samples per subtask (RULER)	64	
Thinking mode	Disabled	N/A
Temperature	0.7	
Top-p	0.95	

Following MemAgent, single-answer tasks use normalized answer matching (Sub-EM), whereas multi-answer tasks are scored by the fraction of target values appearing in the prediction; LongBench v2 follows its official evaluation protocol. The evaluation temperature and top-p also follow MemAgent's official code. Each model reported in this work is trained in a single run, and each evaluation sample is evaluated once per method.

YaRN. We use a scaling factor of 4.0 for positional-encoding extrapolation.

RAG. For each evaluation sample, we construct an Okapi BM25 index over its decoded text chunks; the index is therefore sample-specific rather than a shared external knowledge base. We use the original question as the retrieval query and select the top-6 chunks in descending order of their BM25 scores.

B.2 Detailed Evaluation Results

Table 7: Length-grouped RULER HQA scores and mean turn counts. Asterisks (*) denote published results; averages use available lengths.

Method	7K	14K	28K	56K	112K	224K	448K	896K	1.8M	3.6M	Avg.
Qwen3.5-35B-A3B											
Vanilla	**87.50**	79.69	82.81	81.25	78.12	73.44	62.50	37.50	34.38	29.69	64.69
YaRN	82.81	81.25	79.69	**82.81**	78.12	68.75	59.38	56.25	29.69	14.06	63.28
RAG	84.38	**82.81**	71.88	67.19	46.88	54.69	51.56	53.12	46.88	43.75	60.31
MemAgent (w/o RL)	78.12	78.12	70.31	71.88	62.50	60.94	62.50	68.75	51.56	39.06	64.37
MemAgent	84.38	81.25	82.81	81.25	79.69	68.75	73.44	71.88	71.88	70.31	76.56
PI-Mem (w/o RL)	78.12	75.00	73.44	68.75	67.19	67.19	71.88	64.06	65.62	67.19	69.84
PI-Mem	**87.50**	**82.81**	**84.38**	**82.81**	**84.38**	**79.69**	**82.81**	**76.56**	**75.00**	**76.56**	**81.25**
PI-Mem (w/o RL) Turns	2.133	2.125	2.148	2.227	2.180	2.227	2.258	2.258	2.188	2.344	2.209
PI-Mem Turns	2.016	2.016	2.047	2.047	2.062	2.078	2.078	2.094	2.094	2.156	2.069
Qwen2.5-7B											
Vanilla	60.94	54.69	50.00	23.44	0.00	0.00	0.00	0.00	0.00	0.00	18.91
YaRN	56.25	62.50	62.50	45.31	34.38	0.00	0.00	0.00	0.00	0.00	26.09
RAG	57.81	46.88	46.88	45.31	39.06	42.19	29.69	43.75	26.56	32.81	41.09
Lychee*	80.47	81.25	**82.03**	81.25	80.47	79.69	75.00	75.78	78.12	N/A	79.34
GRU-Mem*	81.25	79.69	77.34	76.56	74.22	73.44	71.88	76.56	N/A	N/A	76.37
ReMemR1*	82.3	82.8	81.1	78.9	82.0	79.7	80.0	80.8	N/A	N/A	80.95
MemAgent (w/o RL)	62.50	59.38	51.56	50.00	43.75	40.62	35.94	37.50	31.25	32.81	44.53
MemAgent	81.25	81.25	75.00	**82.81**	76.56	75.00	76.56	75.00	78.12	73.44	77.50
PI-Mem (w/o RL)	57.03	51.56	47.66	44.53	41.41	46.46	34.38	35.94	35.94	28.12	42.30
PI-Mem	**82.81**	**82.81**	81.25	**82.81**	**89.06**	**85.94**	**82.81**	**84.38**	**87.50**	**81.25**	**84.06**
PI-Mem (w/o RL) Turns	2.883	2.875	2.859	2.875	2.859	2.921	2.930	2.938	2.891	2.906	2.894
PI-Mem Turns	2.312	2.203	2.156	2.219	2.172	2.188	2.234	2.234	2.234	2.328	2.228

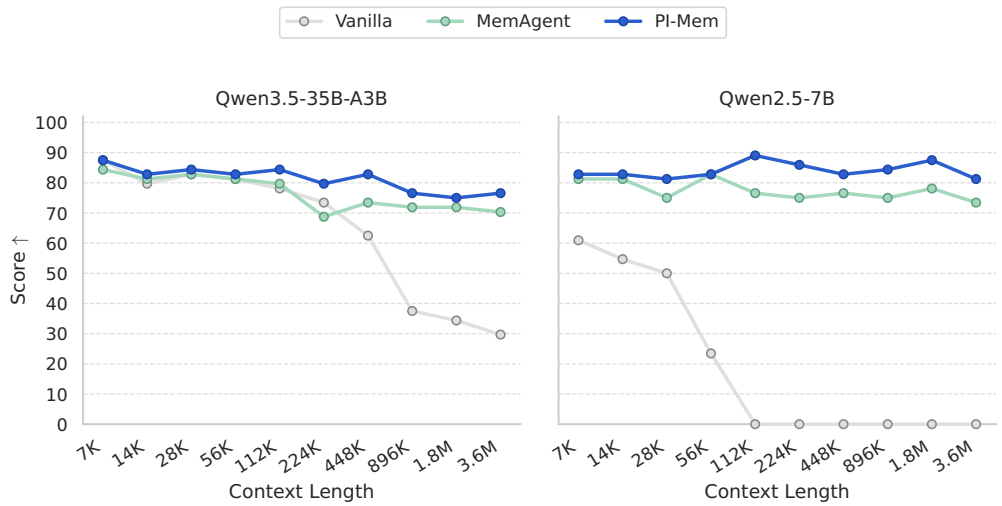


Figure 7: HQA performance across context lengths for Qwen3.5-35B-A3B (left) and Qwen2.5-7B (right).

Table 8: Length-grouped RULER out-of-distribution results. Scores are reported across context lengths from 8K to 1M tokens; mean turn counts are additionally averaged over the 11 out-of-distribution tasks at each length.

Method	8K	16K	32K	64K	128K	256K	512K	1M	Avg.
Qwen3.5-35B-A3B									
Vanilla	98.67	98.39	97.64	97.96	97.40	96.08	95.81	75.56	94.69
YaRN	97.92	96.59	96.74	97.02	97.46	97.22	96.19	86.58	95.72
RAG	98.31	93.07	94.32	95.18	93.25	87.02	77.90	75.24	89.29
MemAgent (w/o RL)	95.50	88.91	77.38	63.93	56.76	50.25	50.03	39.32	65.26
MemAgent	94.55	92.92	95.18	94.05	92.46	90.84	86.42	83.86	91.28
PI-Mem (w/o RL)	97.03	95.66	96.87	96.86	96.43	96.27	95.71	95.15	96.25
PI-Mem	**98.92**	**98.65**	**98.06**	**98.42**	**97.69**	**98.38**	**97.30**	**96.88**	**98.04**
PI-Mem (w/o RL) Turns	2.043	2.071	2.163	2.204	2.212	2.225	2.274	2.294	2.186
PI-Mem Turns	2.055	2.044	2.041	2.021	2.024	2.037	2.065	2.068	2.044
Qwen2.5-7B									
Vanilla	86.57	87.17	84.48	60.65	22.72	0.00	0.00	0.00	42.70
YaRN	84.24	83.03	78.82	72.48	57.13	0.00	0.00	0.00	46.96
RAG	80.58	79.30	77.57	76.49	74.07	68.05	63.45	58.70	72.28
MemAgent (w/o RL)	**94.22**	**91.92**	88.35	85.96	81.16	78.19	72.22	62.41	81.80
MemAgent	91.66	90.33	89.31	87.73	83.73	80.30	79.51	74.83	84.68
PI-Mem (w/o RL)	91.47	88.14	82.66	79.30	77.32	71.99	67.80	60.29	77.37
PI-Mem	91.72	91.78	**92.10**	**92.49**	**91.75**	**91.38**	**89.14**	**88.39**	**91.09**
PI-Mem (w/o RL) Turns	2.825	2.852	2.864	2.905	2.891	2.896	2.920	2.920	2.884
PI-Mem Turns	2.038	2.057	2.057	2.038	2.044	2.065	2.098	2.101	2.062

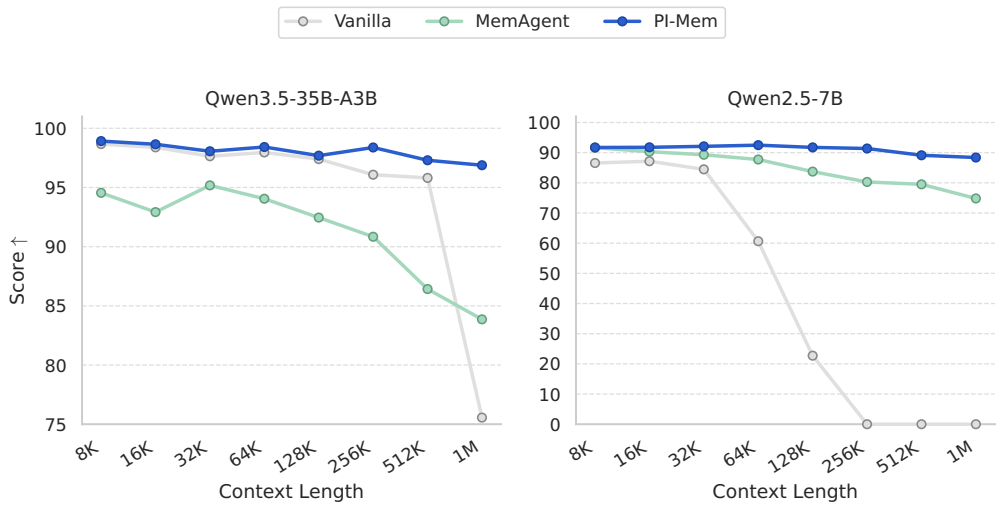


Figure 8: RULER OOD performance across context lengths for Qwen3.5-35B-A3B (left) and Qwen2.5-7B (right).

Table 9: LongBench v2 results using Qwen3.5-35B-A3B. Following the grouping defined by the benchmark, we report results by difficulty (Easy and Hard) and context length (Short, Medium, and Long).

Method	**All**	**Difficulty**		**Length**		
		Easy	Hard	Short	Medium	Long
Vanilla	50.7	55.7	47.6	56.1	47.9	47.2
YaRN	51.5	52.1	**51.1**	**57.8**	48.4	47.2
RAG	48.9	53.6	46.0	51.7	44.7	52.8
MemAgent (w/o RL)	49.3	55.2	45.7	53.9	46.0	48.1
MemAgent	52.3	59.9	47.6	55.6	48.8	53.7
PI-Mem (w/o RL)	51.7	63.5	44.4	51.7	48.4	58.3
PI-Mem	**54.1**	**66.1**	46.6	53.3	**49.8**	**63.9**

B.3 Computational Cost and End-to-End Latency

Table 11 compares Vanilla, MemAgent, and PI-Mem on the 64-sample HQA evaluation sets from 112K to 3.6M tokens. We report analytically estimated algorithmic FLOPs, counting one multiply–accumulate operation (MAC) as two FLOPs; PFLOPs denotes the aggregate operation count in units of 10^{15} FLOPs rather than throughput. For PI-Mem, the estimate includes all chunk reads, refinement turns, memory updates, and final-answer generation, thereby covering the complete inference workload. Wall-clock latency is measured end-to-end for the same 64 samples under the evaluation configuration in Table 6. The speedup ratio is computed as the Vanilla latency divided by the latency of each method.

For a workflow-level asymptotic comparison, let N denote the number of chunks, C the chunk size, M the fixed memory length, and K the maximum number of refinement turns; the total input length is therefore NC. Table 10 reports the dominant chunk-reading attention cost under standard quadratic self-attention. The comparison omits merge and final-answer costs to isolate the effect of workflow structure; these components are included in the complete PFLOPs estimates in Table 11.

Table 10: Asymptotic chunk-reading attention cost and serial depth under quadratic self-attention. Vanilla uses one ultra-long call, whereas PI-Mem processes all N chunk reads in parallel within each turn.

Method	**Attention FLOPs**	**Serial Depth**
Vanilla	$\mathcal{O}((NC)^2)$	1
MemAgent	$\mathcal{O}(N(C+M)^2)$	$\mathcal{O}(N)$
PI-Mem	$\mathcal{O}(KN(C+M)^2)$	$\mathcal{O}(K)$

The asymptotic analysis exposes a compute–latency trade-off for PI-Mem: relative to recurrent memory, it performs up to K times more chunk-reading work but reduces the sequential critical path from $\mathcal{O}(N)$ to $\mathcal{O}(K)$. Because K is bounded and typically much smaller than N for ultra-long inputs, PI-Mem converts additional computation into parallelizable work. The empirical results indicate that this trade-off is most favorable in the ultra-long-context regime, where increased parallelism outweighs the additional computation.

Table 11: Aggregate algorithmic FLOPs and end-to-end wall-clock latency for 64 HQA samples. PFLOPs denotes 10^{15} floating-point operations. FLOPs are analytically estimated, while latency is measured. Speedup is relative to Vanilla.

Length	**Method**	***Qwen3.5-35B-A3B***			***Qwen2.5-7B***		
		PFLOPs ↓	**Latency (s) ↓**	**Speedup ↑**	**PFLOPs ↓**	**Latency (s) ↓**	**Speedup ↑**
112K	Vanilla	103.98	62.94	1.00×	259.70	80.41	1.00×
	MemAgent	57.53	258.89	0.24×	129.70	124.58	0.65×
	PI-Mem	101.01	151.72	0.41×	226.46	80.12	1.00×
224K	Vanilla	344.67	133.17	1.00×	854.33	319.69	1.00×
	MemAgent	121.94	627.33	0.21×	267.99	300.10	1.07×
	PI-Mem	201.07	164.14	0.81×	454.48	154.72	2.07×
448K	Vanilla	1,232.93	416.74	1.00×	3,040.02	849.50	1.00×
	MemAgent	249.47	1,243.47	0.34×	543.88	629.07	1.35×
	PI-Mem	401.34	317.06	1.31×	909.09	309.65	2.74×
896K	Vanilla	4,638.38	1,341.35	1.00×	11,404.64	3,176.89	1.00×
	MemAgent	506.60	2,793.83	0.48×	1,099.70	1,308.64	2.43×
	PI-Mem	806.86	500.39	2.68×	1,832.27	625.76	5.08×
1.8M	Vanilla	18,041.40	13,237.88	1.00×	44,280.96	10,112.76	1.00×
	MemAgent	1,020.13	5,848.31	2.26×	2,215.06	2,634.93	3.84×
	PI-Mem	1,615.29	984.08	13.45×	3,671.00	1,285.31	7.87×
3.6M	Vanilla	71,573.35	33,365.14	1.00×	175,515.76	38,653.00	1.00×
	MemAgent	2,054.65	12,060.71	2.77×	4,462.91	5,535.50	6.98×
	PI-Mem	3,365.35	1,974.65	16.90×	7,633.65	2,682.48	14.41×

C Additional Ablations

C.1 Maximum Turn Budget

Table 12: Effect of the maximum turn budget K on Qwen2.5-7B HQA performance and the mean number of turns used. Bold values denote the best HQA score at each context length; ties are all highlighted.

K	Metric	7K	14K	28K	56K	112K	224K	448K	896K	1.8M	3.6M
1	HQA score	73.44	71.88	75.00	75.00	78.12	79.69	67.19	76.56	67.19	70.31
	Mean turns used	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
3	HQA score	82.81	82.81	**81.25**	**82.81**	89.06	**85.94**	82.81	**84.38**	**87.50**	**81.25**
	Mean turns used	2.312	2.203	2.156	2.219	2.172	2.188	2.234	2.234	2.234	2.328
5	HQA score	**84.38**	**84.38**	79.69	79.69	**90.62**	**85.94**	**84.38**	**84.38**	85.94	**81.25**
	Mean turns used	2.375	2.344	2.109	2.281	2.312	2.234	2.359	2.234	2.375	2.484

Table 12 shows that limiting PI-Mem to one turn consistently reduces HQA accuracy. With $K = 1$, all chunk reads are conditioned only on the initial empty memory, so evidence discovered in one chunk cannot guide subsequent reads of other chunks; iterative refinement is therefore important for cross-chunk information exchange. Increasing the budget from $K = 3$ to $K = 5$, however, does not yield consistent further gains because most samples converge early: under $K = 5$, 75.94% of samples exit after two turns, whereas only 4.84% use four or five turns. PI-Mem thus benefits from iteration while invoking additional turns only for the small subset of samples that require further refinement.

C.2 Chunk and Memory Sizes

We evaluate PI-Mem with Qwen3.5-35B-A3B on HQA while varying one size parameter at a time. The default configuration uses 15K-token chunks and a 4K-token memory. For the chunk-size ablation, the memory size remains 4K; for the memory-size ablation, the chunk size remains 15K.

Table 13: Effect of chunk and memory sizes on PI-Mem with Qwen3.5-35B-A3B on HQA. One parameter is varied at a time, and the shaded row denotes the default 15K/4K configuration.

Chunk	Memory	7K	14K	28K	56K	112K	224K	448K	896K	1.8M	3.6M
5K	4K	71.88	70.31	65.62	70.31	67.19	70.31	62.50	67.19	65.62	65.62
15K	2K	78.12	76.56	67.19	68.75	59.38	65.62	64.06	70.31	65.62	59.38
15K	4K	78.12	75.00	73.44	68.75	67.19	67.19	71.88	64.06	65.62	67.19
15K	8K	78.12	76.56	70.31	70.31	65.62	64.06	70.31	68.75	60.94	67.19
25K	4K	78.12	73.44	67.19	75.00	70.31	67.19	64.06	71.88	62.50	67.19

Table 13 shows no consistent degradation across context lengths when the chunk size varies from 5K to 25K or the memory size varies from 2K to 8K. PI-Mem is therefore robust to these size choices, and its performance is not materially affected within the tested ranges. We do not conduct an additional size ablation for Qwen2.5-7B because its chunk and memory sizes exactly follow the experimental setup of MemAgent.

D PROMPT TEMPLATE

Prompt Template for ReadCall

You are presented with a problem, a section of an article, and a global memory summarizing previously gathered information. Please read the section carefully and determine whether the section contains new information relevant to answering the problem beyond what is already in the global memory. First, output your judgment in the format <**check**>yes</**check**> if there is new information, or <**check**>no</**check**> if there is none. Then, if there is new information, extract and list the key details.

<**problem**>
{prompt}
</**problem**>

<**memory**>
{memory}
</**memory**>

<**section**>
{chunk}
</**section**>

Your response:

Prompt Template for MergeCall

You are presented with a problem and key information extracted from multiple sections of an article. Please consolidate all the information into a single comprehensive memory. Remove redundancies and organize the information clearly, retaining all details relevant to answering the problem.

<**problem**>
{prompt}
</**problem**>

<**extracted_information**>
{memories}
</**extracted_information**>

Consolidated memory:

Prompt Template for FinalCall

You are presented with a problem and a previous memory. Please answer the problem based on the previous memory and put the answer in \boxed{{}}.

<**problem**>
{prompt}
</**problem**>

<**memory**>
{memory}
</**memory**>

Your answer:

E CASE STUDIES

We present paired Qwen3.5-35B-A3B trajectories on the same questions. The MV-NIAH example has 130,778 tokens, and the HQA example has 1,600 documents and 226,114 tokens. We retain the terminology used in the main paper: one PI-Mem *turn* is a complete read–select–merge cycle. Inside the colored boxes, all prose is transcribed from the corresponding JSONL fields; we add only structural labels and color highlighting. Ellipses mark omitted portions of long outputs.

E.1 CASE 1: MEMAGENT ON MV-NIAH

MemAgent on MV-NIAH

Question: Some special magic numbers are hidden within the following text. Make sure to memorize it. What are all the special magic numbers for `cloudy-singing` mentioned in the provided text?

Chunk 0: "One of the special magic numbers for cloudy-singing is: 2952447. July 2010What hard liquor, cigarettes, heroin, and crack have in common is that they're all more concentrated forms of less addictive predecessors. Most if not all the things we describe as addictive are. And the scary thing is, the process that created them is accelerating."

Memory 0: "Based on the provided text, here is the updated memory containing the special magic number for 'cloudy-singing' and other relevant details from the article:
Updated Memory:

- **Special Magic Number for Cloudy-Singing:** 2952447
- **Context of the Number:** The number appears in the text immediately following the phrase 'One of the special magic numbers for cloudy-singing is:' and is followed by the date 'July 2010'.
- **Article Themes & Key Concepts:**
  - **Addictiveness & Technological Progress:** The text discusses how technological progress accelerates the concentration of things we like (e.g., opium to heroin, food processing, social media), making the world more addictive. It notes that social customs (antibodies) evolve slowly compared to technological progress, creating a need for individuals to figure out what to avoid on their own.
  - …

"

…

Chunk 2: "One of the special magic numbers for cloudy-singing is: 8372717."

Memory 2: "Based on the provided text, here is the updated memory containing the new special magic number for 'cloudy-singing' and all other relevant details from the article:
Updated Memory:

- **Special Magic Number for Cloudy-Singing:** 8372717
  - *Note:* **The previous memory listed 2952447.** The text explicitly states: 'One of the special magic numbers for cloudy-singing is: 8372717.' This new number appears in the section discussing how technology makes mass-produced goods (like cars and watches) convenient and affordable for the rich, contrasting them with expensive, custom-made items that are often inferior in performance.
- **Context of the Number:** The number appears in the essay 'Make Money as a Specialized Skill' (April 2004), specifically within a paragraph comparing expensive custom cars to mass-produced cars, and expensive mechanical watches to accurate quartz watches.

…"

Chunk 6: "One of the special magic numbers for cloudy-singing is: 8023581."

Memory 6: "Based on the provided text, here is the updated memory containing the new special magic number for 'cloudy-singing' and all other relevant details from the article:
Updated Memory:

- **Special Magic Number for Cloudy-Singing:** 8023581
  - *Note:* **The previous memory listed 8372717 (and originally 2952447).** The text explicitly states: 'One of the special magic numbers for cloudy-singing is: 8023581.' This new number appears in the section discussing 'Rounds' and the structure of angel rounds versus series A rounds.
- **Context of the Number:** The number appears in the essay 'How to Start a Startup' (April 2012), specifically within a paragraph comparing the equity dilution of traditional Series A rounds (30–40%) with the lower dilution of angel rounds (often around 13%).

...
Correction to Previous Memory:

- The previous memory listed **8372717** as the number for 'cloudy-singing' found in the 'Make Money as a Specialized Skill' essay.
- The text provided in this section explicitly introduces a **new** number: **8023581**.
- *Action:* The memory has been updated to reflect the **most recent explicit statement** regarding the 'special magic number for cloudy-singing' found in the current text, while retaining the context of the previous number for completeness.

"

Chunk 7: "One of the special magic numbers for cloudy-singing is: 2188606."

Memory 7: "Based on the provided text, here is the updated memory containing the new special magic number for 'cloudy-singing' and all other relevant details from the article:
Updated Memory:

- **Special Magic Number for Cloudy-Singing:** 2188606
  - *Note:* **The previous memory listed 8023581 (and originally 8372717 and 2952447).** The text explicitly states: 'One of the special magic numbers for cloudy-singing is: 2188606.'
- **Context of the Number:** The number appears in the essay 'Wisdom and Intelligence' (February 2007), within a discussion distinguishing between 'wise' and 'smart'.

...
Correction to Previous Memory:

- The previous memory listed **8023581** as the number for 'cloudy-singing' found in the 'How to Start a Startup' essay.
- The text provided in this section explicitly introduces a **new** number: **2188606**.
- *Action:* The memory has been updated to reflect the **most recent explicit statement** regarding the 'special magic number for cloudy-singing' found in the current text, while retaining the context of the previous numbers for completeness. The number **2188606 is now the primary reference for 'cloudy-singing'** based on the latest provided text.

"

...

Final Output: "Based on the provided memory, the text explicitly states that the special magic number for 'cloudy-singing' found in the essay 'Wisdom and Intelligence' (February 2007) is **2188606**.
Although the memory notes that previous numbers (8023581, 8372717, 2952447) were associated with 'cloudy-singing' in other contexts or previous updates, the **most recent and explicit statement** in the current text identifies **2188606** as the number.
\boxed{2188606}"

Prediction: 2188606 **Sub-EM:** 0.25
Gold Answer: 2188606, 8023581, 8372717, 2952447

In Memories 2, 6, and 7, each new value replaces the previous value in the main "Special Magic Number" entry, while earlier values are moved to notes or corrections. As a result, although all values remain in memory, the final response returns only the latest one.

E.2 Case 2: PI-Mem on the Same MV-NIAH Sample

PI-Mem on MV-NIAH — Turn 1

Question: Some special magic numbers are hidden within the following text. Make sure to memorize it. What are all the special magic numbers for `cloudy-singing` mentioned in the provided text?

Chunk 0: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, it explicitly states a 'special magic number for cloudy-singing' which is **2952447**. This information was not present in the empty global memory.
Key Details Extracted:

- **Special Magic Number for Cloudy-Singing:** 2952447
- **Context:** The number is mentioned at the very beginning of the provided text section, followed by the date 'July 2010'.
- **Source:** The text appears to be an excerpt from an article (likely by Paul Graham, based on the style and content regarding startups, Lisp, and Viaweb) where this specific number is introduced as a 'special magic number'.

"

Chunk 1: <check>no</check>

Chunk 2: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, it explicitly states a 'special magic number for cloudy-singing' within the text.
Key Details Extracted:

- **Context:** The number is mentioned in a paragraph discussing the difference between mass-produced cars and custom-made cars, and how technology has made the rich live more like the average person.
- **The Number:** **8372717**
- **The Phrase:** 'One of the special magic numbers for cloudy-singing is: 8372717.'

"

Chunk 3: <check>no</check>

Chunk 4: <check>no</check>

Chunk 5: <check>no</check>

Chunk 6: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, it explicitly states a 'special magic number' associated with the phrase 'cloudy-singing' within the text provided.
Key Details Extracted:

- **Context:** The text discusses startup funding, specifically the difference between angel rounds and Series A rounds, and the role of super-angels vs. VCs.
- **The Magic Number:** The text explicitly states: 'One of the special magic numbers for cloudy-singing is: **8023581**.'
- **Location:** This sentence appears in the section titled 'April 2010' (under the sub-section 'Rounds'), embedded within a paragraph discussing equity dilution in Series A rounds.

"

Chunk 7: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, it explicitly states a 'special magic number for cloudy-singing' within the text discussing the relationship between wisdom and intelligence.
Key Details:

- **Context:** The number appears in the essay section titled 'A few days ago I finally figured out something I've wondered about for 25 years: the relationship between wisdom and intelligence.'
- **The Number:** **2188606**
- **The Phrase:** 'One of the special magic numbers for cloudy-singing is: 2188606.'

”

Chunk 8: <**check**>**no**</**check**>

Merged Memory:
“Based on the provided extracted information, here is the consolidated memory regarding the special magic numbers for ‘cloudy-singing’:
Special Magic Numbers for Cloudy-Singing
The text explicitly identifies **four distinct numbers** associated with the phrase ‘cloudy-singing’:

1. **2952447** . . .
2. **8372717** . . .
3. **8023581** . . .
4. **2188606** . . .

”

PI-Mem on MV-NIAH — Turn 2

Chunk 0: <**check**>**no**</**check**>
Chunk 1: <**check**>**no**</**check**>
Chunk 2: <**check**>**no**</**check**>
Chunk 3: <**check**>**no**</**check**>
Chunk 4: <**check**>**no**</**check**>
Chunk 5: <**check**>**no**</**check**>
Chunk 6: <**check**>**no**</**check**>
Chunk 7: <**check**>**no**</**check**>
Chunk 8: <**check**>**no**</**check**>

Merged Memory:
“Based on the provided extracted information, here is the consolidated memory regarding the special magic numbers for ‘cloudy-singing’:
Special Magic Numbers for Cloudy-Singing
The text explicitly identifies four distinct numbers associated with the phrase ‘cloudy-singing’:

1. **2952447** . . .
2. **8372717** . . .
3. **8023581** . . .
4. **2188606** . . .

”

Final Output:
“Based on the provided memory, the special magic numbers for ‘cloudy-singing’ are explicitly listed as:

1. 2952447
2. 8372717
3. 8023581
4. 2188606

The question asks for all the special magic numbers mentioned.
\boxed{2952447, 8372717, 8023581, 2188606}”

Prediction: 2952447, 8372717, 8023581, 2188606 **Sub-EM:** 1.00
Gold Answer: 2188606, 8023581, 8372717, 2952447

Turn 1 extracts four positive chunk memories and merges them as four distinct members of the same set. In Turn 2, every chunk returns <check>no</check>; the consolidated set is preserved for the final call.

E.3 CASE 3: MEMAGENT ON HQA

MemAgent on HQA

Question: Which song was behind the third single on the album "A Lot About Livin' (And a Little 'bout Love)" for two weeks on the country charts?

Memory 0:
"Based on the provided text, there is no information regarding the album "A Lot About Livin' (And a Little 'bout Love)" or the specific song that was the third single on that album and spent two weeks on the country charts. The provided documents contain information about various artists, albums, and events (such as Buddy Holly's "That'll Be the Day", Selena's career, The Chieftains' "The Chieftains in China", and others), but none mention the specific album or single in question.
Therefore, the memory cannot be updated with the answer to the problem as the necessary information is not present in the source text.
Updated Memory:
No previous memory. The provided text does not contain information about the album "A Lot About Livin' (And a Little 'bout Love)" or the third single from that album that spent two weeks on the country charts. The text includes information on other musical topics such as Buddy Holly's final album "That'll Be the Day", Selena's career, The Chieftains' album "The Chieftains in China", and various other unrelated subjects, but none address the specific query."

...

Memory 3:
"Based on the provided text, the memory has been updated with the specific information regarding the album "A Lot About Livin' (And a Little 'bout Love)" and its singles.
Updated Memory:
The provided text now contains specific information regarding the album "A Lot About Livin' (And a Little 'bout Love)" and its singles, which was previously absent.

- **Album:** "A Lot About Livin' (And a Little 'bout Love)" is an album by **Alan Jackson**.
- **First Single:** The first single from this album was **"She's Got the Rhythm (And I Got the Blues)"**, released in October 1992. This song co-written by Alan Jackson and Randy Travis, won an award in 1993 from Music City News for being one of the most performed country songs of the year.
- **Third Single:** The text explicitly states that "She's Got the Rhythm (And I Got the Blues)" was the **first single**. However, the text **does not explicitly name the third single** nor does it explicitly state which song spent "two weeks" on the country charts.

...
Revised Summary of Findings:

- **Album:** "A Lot About Livin' (And a Little 'bout Love)" by Alan Jackson.
- **First Single:** "She's Got the Rhythm (And I Got the Blues)" (Released Oct 1992).
- **Third Single Information:** The provided text **does not identify the third single or its chart performance** (two weeks on the country charts).

Conclusion: The provided text does not contain the answer to the problem. It identifies the album and the first single, but fails to mention the third single or its specific chart duration."

Memory 6:
"Based on the provided text, the memory has been updated with specific information regarding the album "A Lot About Livin' (And a Little 'bout Love)" and its singles.
Updated Memory:

- **Album:** "A Lot About Livin' (And a Little 'bout Love)" is an album by **Alan Jackson**.
- **First Single:** The first single from this album was **"She's Got the Rhythm (And I Got the Blues)"**, released in October 1992.
- **Third Single Information:** The provided text **does not explicitly name the third single** nor does it explicitly state which song spent "two weeks" on the country charts as the third single.

...
Conclusion: The provided text does not contain the answer to the problem. It identifies the album and the first single, but fails to mention the third single or its specific chart duration.
Note: Document 730 mentions a song "We'll Burn That Bridge" by Brooks & Dunn that peaked at #2 for two weeks behind "Chattahoochee" by Alan Jackson, but this does not directly answer the question regarding the third single on the specific album "A Lot About Livin' (And a Little 'bout Love)"."

...

Memory 9:
"**Updated Memory:** ...

- **Fifth/Final Single:** "(Who Says) You Can't Have It All".
- **Third Single Information:** The provided text **does not identify** the third single or its chart performance.

...
Conclusion: The provided text does not contain the answer to the problem. It identifies the album, the first single, and the fifth single, but fails to mention the third single or its specific chart duration.
Note: Document 730 mentions a song "We'll Burn That Bridge" by Brooks & Dunn that peaked at #2 for two weeks behind "Chattahoochee" by Alan Jackson, but this does not directly answer the question regarding the third single on the specific album "A Lot About Livin' (And a Little 'bout Love)"."

Memory 10:
"Based on the provided text, the memory has been updated with specific information regarding the album "A Lot About Livin' (And a Little 'bout Love)" and its singles.
Updated Memory:

- **Album:** "A Lot About Livin' (And a Little 'bout Love)" is the third studio album by American country music artist **Alan Jackson**. It was released on **October 6, 1992**.
- **Singles List:**
  1. **"She's Got the Rhythm (And I Got the Blues)"**
  2. **"Chattahoochee"**
  3. **"Tonight I Climbed the Wall"**
  4. **"(Who Says) You Can't Have It All"**
  5. **"Mercury Blues"**
- **Chart Performance:** "Chattahoochee" and "She's Got the Rhythm (And I Got the Blues)" were both **Number One** hits. The other three songs reached the **Top Five**.

...
Revised Summary of Findings:

- **First Single:** "She's Got the Rhythm (And I Got the Blues)" (Number One).
- **Second Single:** "Chattahoochee" (Number One).
- **Third Single:** "Tonight I Climbed the Wall" (Top Five).
- **Fourth Single:** "(Who Says) You Can't Have It All" (Top Five).
- **Fifth Single:** "Mercury Blues" (Top Five).
- **Answer to Problem:** The text identifies the third single as "Tonight I Climbed the Wall". While the text confirms it reached the Top Five, it does not explicitly confirm the "two weeks" duration mentioned in the problem statement.

Note: The previous memory noted the absence of the third single's name. This update resolves that gap by identifying "Tonight I Climbed the Wall" as the third single based on the sequence provided in Document 1131."

Memory 11:
"Based on the provided text, the memory has been updated with specific information regarding the album "A Lot About Livin' (And a Little 'bout Love)" and its singles, correcting a previous error regarding the release order and chart performance of the third single.
Updated Memory: ...

- **Second Single:** "Chattahoochee" (Number One).
- **Third Single:** "Tonight I Climbed the Wall" (Peaked at Number 4).

- **Answer to Problem:** The text identifies the third single as "Tonight I Climbed the Wall". While the text confirms it reached the Top Five (specifically Number 4), it **does not explicitly confirm** the "two weeks" duration mentioned in the problem statement.

Note: The previous memory incorrectly identified the order of the first two singles and lacked the specific peak position for the third single. This update corrects the release order and adds the specific peak chart position of "Tonight I Climbed the Wall" based on Document 1273."

Memory 12:
"Based on the provided text, the memory has been updated with specific information regarding the album "A Lot About Livin' (And a Little 'bout Love)" and its singles, correcting a previous error regarding the release order and chart performance of the third single.
Updated Memory: ...

- **First Single:** "She's Got the Rhythm (And I Got the Blues)" (Number One).
- **Second Single:** "Tonight I Climbed the Wall" (Number 4).
- **Third Single:** "Chattahoochee" (Number One).
- **Answer to Problem:** The text identifies the third single as "Chattahoochee". While the text confirms it reached Number One, it **does not explicitly confirm** the "two weeks" duration mentioned in the problem statement.

Note: The previous memory incorrectly identified "Tonight I Climbed the Wall" as the third single. Document 1348 corrects this, identifying "Chattahoochee" as the third single released in May 1993."

...

Memory 15:
"Based on the provided text, the memory has been updated with specific information regarding the album "A Lot About Livin' (And a Little 'bout Love)" and its singles, correcting a previous error regarding the release order and chart performance of the third single.
Updated Memory:

- **Album:** "A Lot About Livin' (And a Little 'bout Love)" is the third studio album by American country music artist **Alan Jackson**.
- **Singles List & Order:**
  1. "She's Got the Rhythm (And I Got the Blues)"
  2. "Tonight I Climbed the Wall"
  3. "Chattahoochee"
  4. "(Who Says) You Can't Have It All"
  5. "Mercury Blues"

...
None of the documents in the provided section (1593–1600) contain information regarding the specific chart duration ("two weeks") of the third single from Alan Jackson's album "A Lot About Livin' (And a Little 'bout Love)". The memory update relies on the previously established context (Document 1348) which identified "Chattahoochee" as the third single, while noting the absence of the specific "two weeks" detail in the current text."

Final Output:
"Based on the provided memory, the album "A Lot About Livin' (And a Little 'bout Love)" by Alan Jackson produced five singles. The memory explicitly corrects a previous error to identify the **third single** as **"Chattahoochee"**, which was released in May 1993.
Although the memory notes that the specific text does not explicitly state the "two weeks" duration, it confirms that "Chattahoochee" was the third single and a Number One hit. Therefore, the song in question is "Chattahoochee".
\boxed{Chattahoochee}"

Prediction: Chattahoochee **Sub-EM:** 0.00
Gold Answer: We'll Burn That Bridge

Memory 6 records the answer-bearing relation from Document 730, and Memory 9 still retains it. Memory 10 rewrites the state around a later album track list: the complete Document 730 relation is absent, while an incorrect third-single interpretation becomes the answer candidate. Memory 12

eventually establishes the correct bridge entity, "Chattahoochee," but the earlier relation needed to map that entity to "We'll Burn That Bridge" is no longer available.

E.4 Case 4: PI-Mem on the Same HQA Sample

PI-Mem on HQA — Turn 1

Question: Which song was behind the third single on the album "A Lot About Livin' (And a Little 'bout Love)" for two weeks on the country charts?

Chunk 0: <check>no</check>

Chunk 1: <check>no</check>

Chunk 2: <check>no</check>

Chunk 3: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, **Document 420** identifies the song "She's Got the Rhythm (And I Got the Blues)" as the first single from the album "A Lot About Livin' (And a Little 'bout Love)" by Alan Jackson. While the problem asks for the *third* single, this document provides the context of the album and its singles, which is necessary to answer the question …
Key Details Extracted:

- **Album:** "A Lot About Livin' (And a Little 'bout Love)"
- **Artist:** Alan Jackson
- **First Single:** "She's Got the Rhythm (And I Got the Blues)" (Released October 1992)
- **Award:** The first single received an award in 1993 from Music City News for being one of the most performed country songs of the year.
- **Note:** The provided text does not explicitly state which song was the *third* single or its chart performance duration, but it establishes the album and the first single.

"

Chunk 4: <check>no</check>

Chunk 5: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, **Document 591** mentions the album "A Lot About Livin' (And a Little 'bout Love)" by Randy Travis. While the problem asks about the third single on this album and its chart performance, the provided text in Document 591 actually discusses a *different* album by Randy Travis ("High Lonesome") and mentions that Randy Travis co-wrote a song for Alan Jackson's album "A Lot About Livin' (And a Little 'bout Love)" in 1992. It does not provide the specific details about the singles on the album itself, nor does it identify the third single or its chart performance. …"

Chunk 6: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, **Document 730** identifies a song titled **"We'll Burn That Bridge"** by Brooks & Dunn. This document explicitly states that "We'll Burn That Bridge" peaked at #2 on the country charts for **two weeks, behind "Chattahoochee"** by Alan Jackson. This matches the specific criteria of the question (a song behind another for two weeks on the country charts), even though the document does not explicitly name the album "A Lot About Livin' (And a Little 'bout Love)" in the text provided. However, given the specificity of the "two weeks" and "behind" details, this is the relevant piece of information found in the text.
Key Details Extracted:

- **Song Title:** We'll Burn That Bridge
- **Artist:** Brooks & Dunn
- **Album:** Hard Workin' Man (Note: The document states this song was the second single from *Hard Workin' Man*, not *A Lot About Livin' (And a Little 'bout Love)*, but it is the only song in the text matching the "peaked at #2 for two weeks" description).
- **Chart Performance:** Peaked at #2 on the country charts for two weeks.
- **Song Behind:** "Chattahoochee" by Alan Jackson.

- **Release Date:** May 1993.
- **Writers:** Don Cook and Ronnie Dunn.

"

Chunk 7: <check>no</check>

Chunk 8: <check>no</check>

Chunk 9: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, **Document 1053** identifies the song "(Who Says) You Can't Have It All" as the **fifth and final single** from Alan Jackson's album "A Lot About Livin' (And a Little 'bout Love)". ...
Key Details Extracted:

- **Album:** "A Lot About Livin' (And a Little 'bout Love)" by Alan Jackson.
- **Song:** "(Who Says) You Can't Have It All".
- **Release Date:** January 1994.
- **Position on Album:** Fifth and final single.
- **Chart Performance:** Peaked at number 4 on the U.S. "Billboard" Hot Country Singles chart and number 11 on the Canadian "RPM" Country Tracks chart.
- **Writers:** Alan Jackson and Jim McBride.
- **Album Certification:** Certified $6\times$ Platinum by the RIAA with over 6,000,000 sales in the US.

"

Chunk 10: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, **Document 1131** provides the tracklist and chart performance details for the album "A Lot About Livin' (And a Little 'bout Love)" by Alan Jackson.
Key Details Extracted:

- **Album:** A Lot About Livin' (And a Little 'bout Love) by Alan Jackson.
- **Release Date:** October 6, 1992.
- **Singles Released:** "Chattahoochee", "She's Got the Rhythm (And I Got the Blues)", "Tonight I Climbed the Wall", "(Who Says) You Can't Have It All", and "Mercury Blues".
- **Chart Performance:** "Chattahoochee" and "She's Got the Rhythm (And I Got the Blues)" were Number One hits. The other three songs reached Top Five.
- **Context for the Problem:** The text lists the singles and their peak positions, but it does not explicitly state which specific song was the **third** single or how many weeks it spent on the charts. The third single listed in the text is "Tonight I Climbed the Wall".

"

Chunk 11: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, **Document 1273** identifies the song "Tonight I Climbed the Wall" by Alan Jackson as the second single from the album "A Lot About Livin' (And a Little 'bout Love)". ...
Key Details Extracted:

- **Album:** "A Lot About Livin' (And a Little 'bout Love)"
- **Artist:** Alan Jackson
- **Second Single:** "Tonight I Climbed the Wall"
- **Release Date of Second Single:** January 1993
- **Chart Performance of Second Single:** Peaked at number 4 on the US "Billboard" Hot Country Singles & Tracks chart and the Canadian "RPM" Country Tracks chart.

"

Chunk 12: <check>yes</check>
"The section contains new information relevant to the problem. Specifically, **Document 1348** identifies the song **"Chattahoochee" as the third single** from the album "A Lot About Livin' (And a Little 'bout Love)" by Alan Jackson. While the problem asks which song was

behind the third single for two weeks on the country charts, this document provides the crucial identity of the third single itself, which is the necessary starting point to answer the question . . . ”

Chunk 13: <**check**>**no**</**check**>

Chunk 14: <**check**>**no**</**check**>

Chunk 15: <**check**>**no**</**check**>

Merged Memory:
“Based on the provided extracted information, here is the consolidated memory regarding the album “A Lot About Livin’ (And a Little ’bout Love)” and its singles:
Album Overview

- **Title:** A Lot About Livin’ (And a Little ’bout Love)
- **Artist:** Alan Jackson
- **Release Date:** October 6, 1992
- **Certification:** Certified $6\times$ Platinum by the RIAA (over 6,000,000 sales in the US).

Singles and Chart Performance
The album released multiple singles with the following details extracted from the documents: . . .
Note: There is a discrepancy in the ordering of singles between documents. Document 420/1273/1348 suggest the order: 1. She’s Got the Rhythm, 2. Tonight I Climbed the Wall, 3. Chattahoochee. Document 1131 lists them in a different order but confirms the titles.
Relevance to the Problem
The problem asks for the song that was **behind the third single** on the album for **two weeks** on the country charts.

- The **third single** is identified as **“Chattahoochee”** (Document 1348).
- **Document 730** states that **“We’ll Burn That Bridge”** by Brooks & Dunn peaked at #2 for two weeks, **behind “Chattahoochee”**.
- While “We’ll Burn That Bridge” is not on Jackson’s album, the question asks “Which song was behind the third single...”, implying the song that was charting *below* the third single (Chattahoochee) at that specific time.
- Therefore, the song in question is **“We’ll Burn That Bridge”**.

”

PI-Mem on HQA — Turn 2

Chunk 0: <**check**>**no**</**check**>
Chunk 1: <**check**>**no**</**check**>
Chunk 2: <**check**>**no**</**check**>
Chunk 3: <**check**>**no**</**check**>
Chunk 4: <**check**>**no**</**check**>
Chunk 5: <**check**>**no**</**check**>
Chunk 6: <**check**>**no**</**check**>
Chunk 7: <**check**>**no**</**check**>
Chunk 8: <**check**>**no**</**check**>
Chunk 9: <**check**>**no**</**check**>
Chunk 10: <**check**>**no**</**check**>
Chunk 11: <**check**>**no**</**check**>
Chunk 12: <**check**>**no**</**check**>
Chunk 13: <**check**>**no**</**check**>
Chunk 14: <**check**>**no**</**check**>
Chunk 15: <**check**>**no**</**check**>

Merged Memory:
“**Relevance to the Problem**
The problem asks for the song that was **behind the third single** on the album for **two weeks** on the country charts.

- The **third single** is identified as **“Chattahoochee”** (Document 1348).

- **Document 730** states that **"We'll Burn That Bridge"** by Brooks & Dunn peaked at #2 for two weeks, **behind "Chattahoochee"**.
- Therefore, the song in question is **"We'll Burn That Bridge"**.

"

Final Output:
"Based on the provided memory, the third single on Alan Jackson's album *A Lot About Livin' (And a Little 'bout Love)* is **"Chattahoochee"** (Document 1348).
The memory explicitly states in Document 730 that the song **"We'll Burn That Bridge"** (by Brooks & Dunn) peaked at number 2 on the country charts for two weeks, **behind "Chattahoochee"**.
Therefore, the song that was behind the third single for two weeks is "We'll Burn That Bridge".
\boxed{We'll Burn That Bridge}"

Prediction: **We'll Burn That Bridge** **Sub-EM:** 1.00
Gold Answer: **We'll Burn That Bridge**

The positive reads in Turn 1 independently preserve Document 730's chart relation and Document 1348's third-single identity. Their merged memory states the complete two-hop chain and cites both documents. Turn 2 contributes no new local memories, so the grounded chain remains unchanged for the final call.

Together, the MV-NIAH and HQA pairs show two manifestations of the same recurrent failure mode: later sequential updates can either demote earlier values to obsolete history or delete an unresolved but essential relation. PI-Mem instead retains complementary chunk-level outputs until they can be consolidated within a turn.