InfoMem: Training Long-Context Memory Agents with Answer-Conditioned Information Gain

Anonymous ACL submission

Abstract

Long-context tasks require LLMs to identify and preserve answer-relevant information from large contexts. Chunk-wise memory agents address this issue by sequentially reading document chunks, updating a compact memory, and generating the final answer from the accumulated memory. However, existing RL-based chunk-wise agents either rely on sparse final-answer rewards or use lexical intermediate rewards for memory and retrieval actions. These signals supervise task success or local overlap, but do not directly evaluate whether the final memory supports the ground-truth answer. We propose InfoMem, a reward mechanism for training chunk-wise memory agents that evaluates final-memory utility using answer-conditioned information. InfoMem measures how much the final memory increases the model's per-token log-likelihood of the ground-truth answer. To stabilize RL optimization, InfoMem applies this signal only to successful trajectories and normalizes it before reward composition. Under the same GRPO framework and training budget, InfoMem improves long-context memory-agent performance over comparable memory-agent RL baselines. Analyses show that effective final-memory rewards should operate on successful trajectories, be normalized before reward composition, and be conditioned on the answer rather than the query.

1 Introduction

Long-context understanding has become a central capability for large language models (LLMs), with applications ranging from long-document question answering to corpus-level evidence aggregation (OpenAI, 2025; Wu et al.; Hsieh et al.; Lu et al., 2026). Prior work has improved long-context processing through extended context windows (Shen et al., 2025), attention or positional modifications (Munkhdalai et al., 2024; Press et al.), retrieval augmentation (Zhao et al., 2024), and memory-based or agentic pipelines (Packer et al., 2024; Zhou et al., 2025). However, effectively using long inputs remains challenging when relevant evidence is sparse, distributed across distant segments, or must be preserved throughout a long reading process.

Among these approaches, chunk-wise memory agents provide a simple and effective paradigm for long-context reasoning. Instead of processing the entire document at once, the model sequentially reads shorter chunks, updates a compact memory state, and generates the final answer from the accumulated memory. This paradigm appears in training-free reading agents (Lee et al., 2024), recurrent or memory-augmented model architectures (Dai et al., 2019), and post-training methods (Yu et al., 2025; Shi et al., 2026). Crucially, their explicit memory state makes memory quality an observable optimization target.

Despite their effectiveness, existing chunk-wise long-context systems still lack a scalable way to supervise memory formation. Training-free methods often rely on human-designed memory-update prompts, summarization heuristics, or fixed traversal workflows (Lee et al., 2024). Architecture-level methods may improve long-context capability more fundamentally (Dai et al., 2019), but typically require costly pretraining. Reinforcement learning (RL)-based memory agents can also improve long-context behavior through task feedback, but existing methods mainly rely on sparse answer rewards (Yu et al., 2025) or lexical intermediate rewards for memory and retrieval actions (Shi et al., 2026). These rewards supervise task success or local word overlap, without directly evaluating whether the final memory semantically supports the ground-truth answer.

This limitation is especially pronounced within successful trajectories: sparse outcome rewards cannot distinguish whether the final memory contains focused answer-supporting evidence or redun-

dant distracting information, while lexical rewards may not capture semantic support for the final answer. This motivates a memory-specific reward signal for chunk-wise long-context reinforcement learning.

We propose InfoMem, a reward-shaping method for chunk-wise long-context memory agents based on answer-conditioned information gain. The core intuition is that a useful final memory should increase the model's support for the ground-truth answer. Instead of estimating distribution-level mutual information, InfoMem uses a point-wise information-gain surrogate by comparing the model's per-token log-likelihood of the ground-truth answer with and without the final memory. InfoMem further improves training stability by applying this signal only to successful trajectories and normalizing it before reward composition.

Experiments show that InfoMem consistently improves long-context memory-agent performance over outcome-only GRPO and a comparable memory-agent RL baseline ReMemR1 (Shi et al., 2026). Further analyses show that effective final-memory rewards should operate on successful trajectories, be normalized before reward composition, and be conditioned on the ground-truth answer rather than on the query alone. These findings suggest that answer-conditioned information gain provides a principled framework for final-memory supervision in chunk-wise long-context RL.

Our contributions are threefold: (1) We formulate final-memory utility through an information-theoretic perspective, where useful memories should reduce the model's uncertainty about the ground-truth answer. (2) We introduce InfoMem, an answer-conditioned information-gain reward for direct final-memory shaping over successful trajectories. (3) We show that InfoMem consistently improves chunk-wise long-context memory agents over comparable memory-agent RL baselines, and further identify three key properties of effective final-memory rewards: success-side supervision, pre-composition normalization, and answer conditioning.

2 Related Work

2.1 Long-context LLMs and Chunk-wise Memory Agents

Long-context LLM research has improved long-input processing through context extension (Shen et al., 2025), attention or positional modifications (Munkhdalai et al., 2024; Press et al.), and retrieval augmentation (Zhao et al., 2024).

Complementary to these approaches, chunk-wise memory agents maintain an explicit memory state during sequential long-document processing. This paradigm includes training-free reading workflows (Lee et al., 2024), segment-level recurrent architectures (Dai et al., 2019; Hutchins et al., 2022; Ding et al., 2021), and post-training memory agents (Yu et al., 2025; Shi et al., 2026). Existing methods rely on manually designed workflows, costly recurrent architectures, or RL objectives based on sparse final-answer rewards and intermediate memory heuristics. As a result, direct supervision of answer-conditioned final-memory utility remains relatively underexplored.

2.2 RL for Long-context QA

DeepSeek-R1 (Guo et al., 2025) demonstrates that reinforcement learning can substantially enhance LLM capabilities in specialized domains. Recent work further shows that RL can improve long-context question answering and reasoning. Early methods mainly optimize verifiable end-task outcomes, such as final-answer correctness or verifier-based response quality (Shen et al., 2025; Yu et al., 2025). More recent studies have introduced denser supervision signals for grounding, evidence extraction, and contextual reasoning (Chen et al.; Guan et al., 2026; Ping et al., 2026; Shi et al., 2026). Together, these results suggest that reward-based optimization is a promising direction for improving long-context reasoning.

Despite these advances, existing rewards mainly supervise grounding quality, evidence selection, reading utility, or intermediate memory-update behavior. ReMemR1 (Shi et al., 2026) introduces information-style rewards for memory and callback actions, but these signals are based on word-level recall rather than answer-conditioned final-memory utility. Direct supervision of the final memory representation itself remains relatively underexplored. In particular, existing methods rarely evaluate whether the resulting final memory directly supports the ground-truth answer, which is the focus of our work.

3 Problem Setup and Motivation

3.1 Chunk-wise Long-context Memory Agent

We focus on chunk-wise memory agents as a practical paradigm for reinforcement learning in long-

context settings. Formally, let x denote the query, D the long document, and y^* the ground-truth answer. Given a pre-defined chunk size C, the document is divided into K chunks:

$$D = \{c_1, c_2, \ldots, c_K\}, \tag{1}$$

Conditioned on the query, the model sequentially reads the chunks and maintains a memory state:

$$M_t = \pi_\theta(M_{t-1}, c_t, x), \quad t = 1, \ldots, K. \tag{2}$$

After processing all chunks, the model obtains the final memory M_K and generates the final answer based on the query and this memory:

$$\hat{y} = \pi_\theta(x, M_K). \tag{3}$$

3.2 Why Outcome Reward Is Insufficient for Memory Learning

The outcome reward directly supervises final answer correctness, but provides only sparse and indirect supervision for memory utility. In chunk-wise memory agents, the final prediction is generated based on the final memory M_K, which is expected to retain the information necessary for supporting the correct answer. However, the binary outcome reward evaluates only whether the generated answer matches y^*, without directly distinguishing the quality of different final memories.

This limitation is particularly evident among successful trajectories. Multiple rollouts may generate the same correct answer and therefore receive identical outcome rewards, while their final memories can differ substantially in utility. Some memories may retain only the critical supporting evidence, whereas others may preserve the same evidence together with substantial redundant or distracting information while still yielding the correct prediction. Consequently, outcome reward alone cannot differentiate memory utility within successful trajectories, motivating the need for a direct reward signal for final-memory utility in chunk-wise long-context reinforcement learning.

3.3 From Mutual Information to Model-induced Pointwise Surrogate

Long-context question answering can be viewed as extracting answer-relevant information from a large context. Under the chunk-wise memory-agent formulation, the final memory should therefore reduce the uncertainty of the answer conditioned on the query. Ideally, this utility can be characterized by the conditional mutual information $I(M; Y \mid X)$, which measures how much additional information the memory M provides about the answer Y given the query X.

However, distribution-level mutual information depends on the full joint distribution over queries, memories, and answers, and is difficult to estimate reliably in the high-dimensional semantic space of LLMs (Qian et al., 2025). This motivates the InfoMem reward introduced in Section 4, where we instantiate the mutual-information objective with a single-sample pointwise surrogate. This surrogate measures whether the final memory increases the model's support for the ground-truth answer on the current instance.

4 Method: InfoMem

4.1 Reward Definition

We train the chunk-wise memory agent with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For each prompt, GRPO samples a group of n rollouts,

$$\mathcal{G} = \{1, 2, \ldots, n\}. \tag{4}$$

Each rollout $i \in \mathcal{G}$ produces a final memory M_i and a final answer $\hat{y}_i$. The base outcome reward is defined as

$$R_{\text{outcome},i} = \mathbb{1}[\hat{y}_i = y^*], \tag{5}$$

where answer correctness is evaluated by normalized string matching in our main experiments.

Given a query x, a final memory M, and the ground-truth answer y^*, we define the answer-conditioned information-gain reward r_{gain} as

$$r_{\text{gain}}(x, M, y^*) = \frac{1}{|y^*|} \log P_\theta(y^* \mid x, M) - \frac{1}{|y^*|} \log P_\theta(y^* \mid x, \emptyset), \tag{6}$$

where $\emptyset$ denotes null memory. The sequence likelihood of LLM on ground-truth tokens P_θ is computed under teacher forcing:

$$\log P_\theta(y^* \mid x, M) = \sum_{j=1}^{|y^*|} \log p_\theta(y_j^* \mid y_{<j}^*, x, M), \tag{7}$$

which corresponds to the summed log-probability of the ground-truth tokens. The first term in Eq. (6) measures the likelihood assigned to the ground-truth answer when conditioned on the final memory, while the second term measures the corresponding

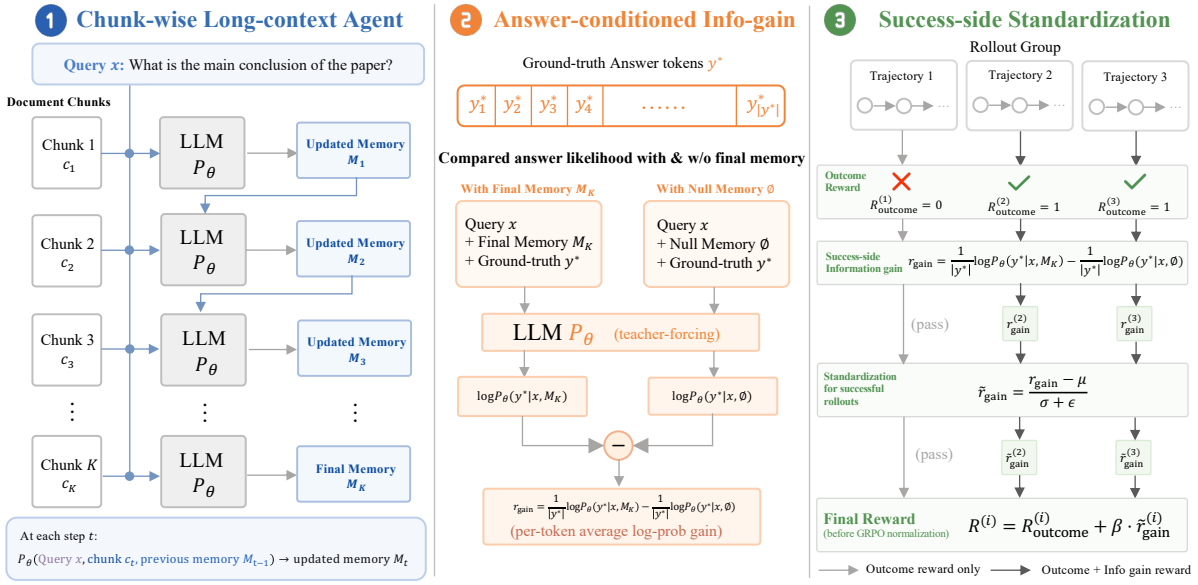


Figure 1: Overview of InfoMem for chunk-wise long-context RL. InfoMem measures final-memory utility using answer-conditioned information gain by comparing the teacher-forced per-token average log-likelihood of the ground-truth answer y^* with and without the final memory. During GRPO training, information-gain supervision is applied only to successful trajectories, normalized across successful rollouts, and combined with the sparse outcome reward for policy optimization.

likelihood without memory conditioning. Consequently, higher values of r_{gain} indicate that the final memory increases the likelihood assigned to the correct answer.

4.2 Using Successful Trajectories as Positive Memory Signals

Given the rollout group $\mathcal{G}$ and the outcome rewards defined above, we define the set of successful rollouts as

$$\mathcal{S} = \{i \in \mathcal{G} : R_{\text{outcome},i} = 1\}. \tag{8}$$

InfoMem applies the information-gain reward only to rollouts in $\mathcal{S}$. Specifically, successful trajectories are further differentiated by $r_{\text{gain}}(x, M_i, y^*)$, whereas failed trajectories are supervised with R_{outcome} only.

This design restricts information-gain optimization to trajectories whose final answers are already validated by the outcome reward. Within this subset, differences in r_{gain} more directly reflect the extent to which the final memory supports the ground-truth answer y^*, thereby providing a more stable signal for memory shaping.

By contrast, failed trajectories may entangle memory quality with incorrect evidence selection or answer generation, making r_{gain} less reliable as a memory-utility signal. We empirically analyze this issue through wrong-only and both-side supervision variants in Section 6.

4.3 Memory Reward Normalization

The scale of raw information-gain rewards can vary substantially across prompts due to differences in answer uncertainty. For relatively easy questions, the model may already assign high likelihood to the ground-truth answer without memory conditioning, resulting in limited likelihood improvement from the final memory. For more difficult questions, an informative final memory can produce a substantially larger increase in ground-truth likelihood. Consequently, directly combining raw r_{gain} with the binary outcome reward may introduce severe reward-scale imbalance, causing the information-gain term to dominate the sparse outcome signal for some prompts while remaining negligible for others.

Following the reward-decoupled normalization setting in GDPO (Liu et al., 2026), InfoMem controls this scale before reward composition by normalizing information-gain values within the successful trajectories of the same rollout group. Let $r_i = r_{\text{gain}}(x, M_i, y^*)$ for $i \in \mathcal{S}$. We compute

$$\tilde{r}_i = \frac{r_i - \mu_{\mathcal{S}}}{\sigma_{\mathcal{S}} + \epsilon}, \quad i \in \mathcal{S}, \tag{9}$$

where $\mu_{\mathcal{S}}$ and $\sigma_{\mathcal{S}}$ denote the mean and standard

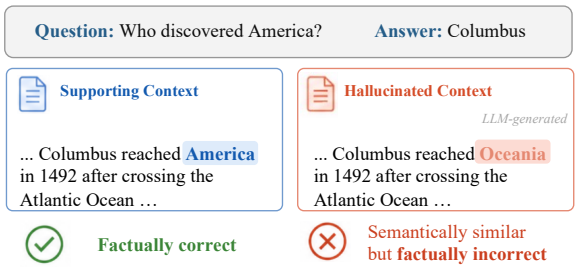


Figure 2: Example of synthetic hallucinated evidence.

Score	MRR	Z-score SNR
Embedding (BGE-M3)	0.719	0.316
Embedding (E5)	0.714	0.320
Embedding (Qwen3-0.6B)	0.727	0.371
ATTN-MASS	0.588	-0.081
ATTN-TOP1	0.792	0.577
r_{gain} (ours)	**0.977**	**2.960**

Table 1: Synthetic context discrimination results. The r_{gain} score corresponds to the answer-conditioned information-gain score in Eq. (6).

deviation of $\{r_i : i \in \mathcal{S}\}$, and ϵ is a small constant for numerical stability. When $|\mathcal{S}| = 0$, no information-gain reward is applied. When $|\mathcal{S}| = 1$, the original value is retained as a fallback strategy.

The final reward for rollout i is defined as

$$R_i = \begin{cases} R_{\text{outcome},i} + \beta \tilde{r}_i, & i \in \mathcal{S}, \\ R_{\text{outcome},i}, & i \notin \mathcal{S}, \end{cases} \tag{10}$$

where β controls the strength of the normalized information-gain term.

This normalization differs from the group-relative normalization in GRPO. InfoMem normalizes r_{gain} before reward composition, controlling the relative scale between the information-gain and outcome rewards.

Although InfoMem is computed from the final memory and final answer, the resulting trajectory-level advantage is propagated to all generated tokens, including both memory-update and final-answer tokens. The complete reward-construction procedure is summarized in Appendix A.2.

5 Experiments

5.1 Synthetic Context Discrimination with Hallucinated Evidence

We first evaluate whether answer-conditioned information gain can distinguish genuinely answer-supporting evidence from surface-similar but factually misleading contexts. We construct a synthetic diagnostic using the SQuAD dataset (Rajpurkar et al., 2016). Each example contains a question, a supporting context, and a ground-truth answer.

For each retained QA pair, we generate two hallucinated contexts using Gemini 3 Flash preview (Google, 2026). The hallucinated contexts preserve the overall structure and semantic style of the original passage while replacing answer-critical facts so that they no longer support the ground-truth answer, as illustrated in Figure 2.

We compare three categories of evidence scores: (1) **Information gain.** We compute the r_{gain} score using the same Qwen2.5-1.5B-Instruct model (Qwen et al., 2025) as in training, comparing the per-token average log-likelihood of the ground-truth answer with and without the candidate context. (2) **Embedding similarity.** We compute cosine similarity between the embedding of candidate-context and question-answer template, using mainstream open-source embedding models (Zhang et al., 2025; Chen et al., 2024; Wang et al., 2024). (3) **Attention-based scores.** Using the same Qwen2.5-1.5B-Instruct model, we evaluate two attention metrics. ATTN-MASS averages the total attention mass assigned from answer tokens to context tokens, while ATTN-TOP1 averages the maximum attention score assigned to any context token for each answer token.

We evaluate each score from two complementary perspectives. First, we compute the mean reciprocal rank (MRR) of the true supporting context among the three candidates. Second, we report a Z-score signal-to-noise ratio (SNR) to assess the stability of this discrimination signal. A higher SNR indicates that the score not only separates true contexts from hallucinated ones, but also does so with lower relative variation across samples, making it more suitable as a training reward.

Table 1 shows that r_{gain} achieves the highest MRR and Z-score SNR among all compared scores. The high MRR indicates strong discrimination between true and hallucinated contexts, while the substantially larger SNR suggests that the resulting signal is also more stable across samples. Embedding-based similarities achieve moderate ranking performance but exhibit much weaker SNR. Attention-based scores, especially ATTN-TOP1, improve ranking quality but remain substantially below r_{gain} in signal stability. These results suggest that answer-conditioned likelihood gain provides both stronger evidence discrimination and a cleaner reward signal for training. Details of the discrimination experiment are provided in Appendix B.

Model	CorpusQA	LongMemEval	MRCR-8needle	RULER synthetic QA
Qwen2.5-1.5B-Instruct	14.590	5.600	0.260	13.308
with Outcome-only GRPO	16.413	10.000	0.063	34.735
with InfoMem	**19.453**	**12.800**	**0.279**	**36.848**
with ReMemR1 (Shi et al., 2026)	1.520	6.200	0.092	28.919

Table 2: Main evaluation results on long-context benchmarks. All scores are reported as percentages. The best score for each benchmark is shown in bold.

5.2 Training Setup

We train on the RULER-HotpotQA dataset (Yu et al., 2025), which applies the RULER long-context construction paradigm (Hsieh et al.) to HotpotQA (Yang et al., 2018) by mixing answer-relevant documents with distractors. Each example contains 200 documents. The original training split has 32,768 examples and the validation split has 128 examples; for controlled experimentation, we downsample the training split to 512 examples and keep the validation split unchanged.

All experiments use Qwen2.5-1.5B-Instruct as the base model and GRPO as the algorithm. For each prompt, we sample $n = 8$ rollouts and train for 120 steps with learning rate 1×10^{-6}. The chunk size is 5000 tokens, the maximum memory length is 1024 tokens, and the InfoMem coefficient is $\beta = 0.2$. For controlled comparison, all methods use the same data, model, rollout number, decoding configuration, and training budget. Our primary baseline is Outcome-only GRPO (Yu et al., 2025), which removes the information-gain term while keeping the same training pipeline. We also report the initial model before RL optimization where appropriate. Since ReMemR1 (Shi et al., 2026) augments the chunk-wise memory-agent framework with callback retrieval, we align all shared training parameters with our setting for fair reproduction, while keeping ReMemR1-specific parameters consistent with the original paper. Details are provided in Appendix A.

5.3 Evaluation

5.3.1 Benchmarks

Under the same chunk-wise memory framework, we evaluate the trained models on a set of out-of-distribution long-context benchmarks covering complementary forms of evidence use: (1) **MRCR-8needle** (OpenAI, 2025) evaluates multi-needle retrieval; (2) **RULER synthetic QA** (Yu et al., 2025) evaluates sparse retrieval-style question answering, using a different corpus source from the training data; (3) **CorpusQA** (Lu et al., 2026) evaluates corpus-level evidence aggregation across document collections; and (4) **LongMemEval** (Wu et al.) evaluates long-horizon dialogue memory and conversational information tracking.

Across benchmarks, we use the metric specified by each benchmark and apply the same evaluation protocol to all compared methods, except that ReMemR1 is evaluated under its original chunk-wise framework with callback retrieval for train-test consistency. CorpusQA and LongMemEval are evaluated with LLM-as-judge using Kimi-K2.6 (Moonshot AI, 2026)[1], MRCR-8needle with sequence match, and RULER synthetic QA with F1 score. For evaluation efficiency, we downsample the original 800 MRCR-8needle examples to 100 examples, use the 128K-token subset of CorpusQA, and use the 115K-token LongMemEval-S subset. Details for evaluation are provided in Appendix C.

5.3.2 Main Results

Table 2 shows that InfoMem achieves the best overall performance across all four long-context benchmarks, outperforming both Outcome-only GRPO and ReMemR1. Compared with the initial model, InfoMem consistently improves both retrieval-oriented tasks and more memory-intensive settings. These results suggest that answer-conditioned information gain provides an effective supervision signal for final-memory formation beyond sparse final-answer correctness.

In contrast, Outcome-only GRPO produces less consistent improvements. Although it improves over the initial model on CorpusQA, LongMemEval, and RULER synthetic QA, it substantially degrades performance on MRCR-8needle, falling below the initial model. This result suggests that optimizing only sparse outcome rewards may lead to degraded long-context retrieval behavior. InfoMem avoids this degradation and achieves the best overall performance, supporting the need for a memory-specific reward signal in chunk-wise

[1] Manual verification shows high similarity to human judgments: 99.5% on CorpusQA and 96.8% on LongMemEval.

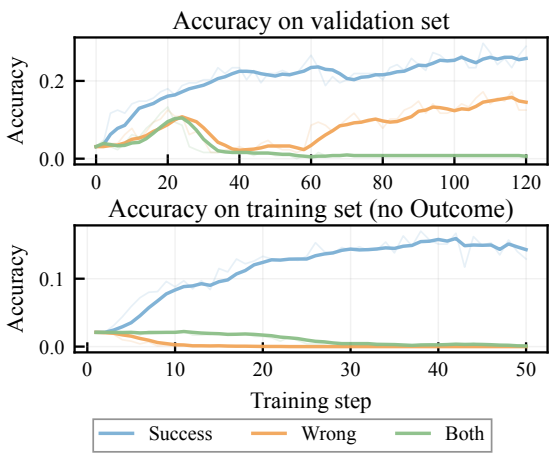


Figure 3: Effect of information-gain supervision side. Top: with outcome reward, validation accuracy measures generalization. Bottom: without explicit outcome reward, training accuracy diagnoses whether each supervision side provides a usable learning signal. Thick lines show sliding-window smoothing with window size 5, and light lines show raw values.

long-context RL.

ReMemR1 also fails to match InfoMem under our sample-aligned evaluation. One possible explanation is that its intermediate reward design is primarily based on word-level recall against the ground-truth answer, which may encourage lexical overlap rather than preserving semantically supporting evidence.

6 Analysis and Ablation

We next analyze the key design choices underlying InfoMem. Specifically, we study which trajectories should receive information-gain supervision, why information-gain rewards must be normalized before reward composition, and why the reward should be conditioned on the ground-truth answer rather than on the query.

6.1 Which Side Should Receive Information-gain Supervision?

We analyze which trajectories should receive information-gain supervision. Specifically, we compare three supervision-side variants: *Success* (the default InfoMem setting); *Wrong*, which applies r_{gain} only to failed trajectories; and *Both*, which applies it to both. All variants use the same training setup, differing only in the supervision.

6.1.1 Main Setting with Outcome Reward

We retain the outcome reward and vary only the supervision side of the information-gain term. As shown in the top panel of Figure 3, success-only supervision produces the most stable validation. This behavior supports the role of successful trajectories as positive memory examples: their final memories are already associated with correct final answers, and r_{gain} can further distinguish which successful memories provide stronger support for the ground-truth answer.

By contrast, wrong-side and both-side supervision exhibit unstable training dynamics. Both variants initially improve but collapse during mid-stage training, with the both-side variant remaining near zero afterward. This suggests that failed trajectories may provide weak useful signals early in training, but their associated memories become increasingly noisy as optimization proceeds. Mixing successful and failed memories within the same reward objective further weakens the positive memory-shaping effect of success-only supervision.

6.1.2 No-outcome Diagnostic

We further conduct a diagnostic experiment without explicit outcome reward. Rather than serving as the main training objective, this experiment tests whether r_{gain} alone can provide a usable learning signal under different supervision sides. Specifically, we remove R_{outcome} and optimize using only the information-gain reward under success-only, wrong-only, and both-side supervision.

The bottom panel of Figure 3 shows that only success-only supervision produces sustained improvement in training accuracy, whereas wrong-only and both-side supervision fail to produce stable learning behavior. This result suggests that success-side information gain can provide meaningful memory shaping even without explicit outcome reward, while wrong-side supervision does not provide a reliable optimization signal. Overall, these findings support using information-gain reward as positive memory shaping over successful trajectories rather than as uniform supervision across all trajectories.

6.2 Why Is Normalization Necessary Before Reward Composition?

We evaluate the role of pre-composition normalization by removing the normalization of r_{gain} before reward composition while keeping all other settings unchanged. As shown in Table 3, removing this

Model	CorpusQA	LongMemEval	MRCR-8needle	RULER synthetic QA
InfoMem	**19.453**	**12.800**	**0.279**	**36.848**
InfoMem w/o r_{gain} normalization	16.109	11.600	0.029	35.371
InfoMem with QueryPMI	18.237	8.000	0.117	26.163

Table 3: Ablation results on long-context benchmarks. All scores are reported as percentages. The best score for each benchmark is shown in bold.

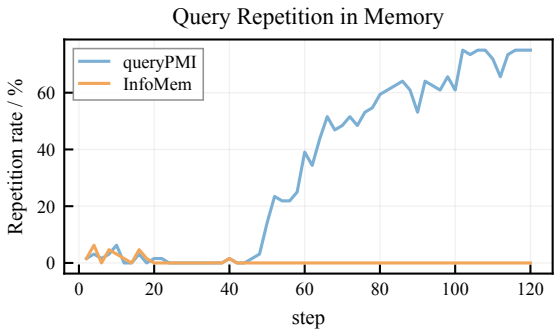


Figure 4: Fraction of rollouts whose final memory repeats the query during training.

normalization consistently degrades performance across all evaluated benchmarks. The degradation is especially pronounced on retrieval-oriented tasks, suggesting that raw likelihood-gain rewards are not directly comparable across rollout groups.

These results support the need for normalization before reward composition. Since raw r_{gain} magnitudes can vary substantially across prompts, directly combining them with sparse outcome rewards may introduce unstable reward scaling. Normalizing r_{gain} within successful trajectories converts it into a relative memory-quality signal before reward composition, producing a more stable reward-shaping signal during training.

6.3 Why Must the Reward Be Answer-conditioned Rather Than Query-conditioned?

We next examine whether query relevance alone is sufficient for final-memory reward design. To this end, we construct a query-conditioned variant, QueryPMI, which replaces the ground-truth answer in r_{gain} with the query itself. Specifically, given a final memory M and the query x, we use the template *"Based on the previous memory {M}, we can answer the query {x}."* and compute the corresponding per-token log-likelihood gain on query. This formulation is intuitively appealing because useful memories are often related to the query. However, query relevance is not equivalent to answer support: a memory can increase query likelihood simply by copying or paraphrasing the query without preserving answer-supporting evidence.

Table 3 shows that QueryPMI performs consistently worse than InfoMem across all evaluated benchmarks, indicating that query-conditioned likelihood gain is a substantially weaker final-memory reward than answer-conditioned information gain. Figure 4 further explains this degradation. During training, QueryPMI rapidly increases the proportion of rollouts whose final memories repeat the query, whereas InfoMem keeps this proportion close to zero. This behavior suggests that the model exploits the query-conditioned objective by making the query easier to predict from memory rather than learning to preserve answer-supporting evidence. These results highlight the importance of conditioning the reward on the ground-truth answer rather than on the query alone. Additional training and validation curves are provided in Appendix E.

7 Conclusion

We studied final-memory reward design for chunk-wise long-context memory agents. While outcome-only GRPO supervises memory formation only indirectly through final-answer correctness, InfoMem introduces answer-conditioned information gain as a direct reward signal for final-memory utility. Experiments show that InfoMem consistently improves long-context memory-agent performance over outcome-only GRPO and a comparable memory-agent RL baseline, ReMemR1.

Further analyses reveal three key properties of effective final-memory rewards: they should operate as positive memory shaping over successful trajectories, be normalized before reward composition, and be conditioned on the ground-truth answer rather than on the query alone. Overall, our results suggest that answer-conditioned information gain provides a more direct and effective supervision signal for learning useful final memories in chunk-wise long-context RL.

Limitations

Despite these promising results, our current study still has several limitations.

First, our experiments use a limited training subset and a relatively small base model. This design reflects both the high computational cost of long-context reinforcement learning and our focus on controlled evaluation of the proposed reward design rather than large-scale benchmark optimization. Scaling InfoMem to larger models and substantially larger training corpora remains an important direction for future study.

Second, this work focuses specifically on chunk-wise long-context memory agents. The proposed reward is designed for settings in which the model sequentially processes context chunks, maintains an explicit memory state, and generates the final answer from the resulting final memory. Its applicability to other long-context paradigms, such as retrieval-only systems or full-context single-pass models, remains unexplored.

Third, the current reward is defined only at the final step. Although GRPO propagates the resulting trajectory-level advantage to all generated tokens, the reward itself evaluates only the final memory and final answer. Extending answer-conditioned information gain toward intermediate memory states and step-wise process supervision remains future work.

Potential risks should also be considered. Since InfoMem encourages final memories that increase support for a target answer, erroneous answers or biased source documents may lead the model to preserve and amplify misleading evidence. The answer-conditioned reward may also be over-optimized toward memories that are highly associated with the expected answer while omitting important qualifications from the original context. These risks are especially relevant in high-stakes long-document applications, such as legal, medical, or financial analysis, where compressed memory states should not replace source-document verification or human review.

References

Guanzheng Chen, Michael Qizhe Shieh, and Lidong Bing. Longrlvr: Long-context reinforcement learning requires verifiable context rewards. In *The Fourteenth International Conference on Learning Representations*.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Siyu Ding, Junyuan Shang, Shuohuan Wang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2021. Erniedoc: A retrospective long-document modeling transformer. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2914–2927.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. The language model evaluation harness.

Google. 2026. Gemini 3 flash preview. `https://ai.google.dev/gemini-api/docs/models/gemini-3-flash-preview`. Google AI for Developers. Last updated 2026-04-28 UTC. Accessed: 2026-05-19.

Xin Guan, Zijian Li, Shen Huang, Pengjun Xie, Jingren Zhou, and Jiuxin Cao. 2026. Evidence-augmented policy optimization with reward co-evolution for long-context reasoning. *Preprint*, arXiv:2601.10306.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

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

DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, and Behnam Neyshabur. 2022. Block-recurrent transformers. In *Advances in Neural Information Processing Systems*, volume 35, pages 33248–33261. Curran Associates, Inc.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving

with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. In *International Conference on Machine Learning*, pages 26396–26415. PMLR.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. 2026. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization. *Preprint*, arXiv:2601.05242.

Zhiyuan Lu, Chenliang Li, Yingcheng Shi, Weizhou Shen, Ming Yan, and Fei Huang. 2026. Corpusqa: A 10 million token benchmark for corpus-level analysis and reasoning. *Preprint*, arXiv:2601.14952.

Moonshot AI. 2026. Kimi k2.6: Advancing open-source coding. Kimi Tech Blog. Accessed: 2026-05-19.

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

OpenAI. 2025. Openai mrcr: Long context multiple needle in a haystack benchmark. `https://huggingface.co/datasets/openai/mrcr`. Hugging Face dataset repository. Accessed: 2026-05-19.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Bowen Ping, Zijun Chen, Yiyao Yu, Tingfeng Hui, Junchi Yan, and Baobao Chang. 2026. Longr: Unleashing long-context reasoning via reinforcement learning with dense utility rewards. *Preprint*, arXiv:2602.05758.

Ofir Press, Noah Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. 2025. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in llm reasoning. *Preprint*, arXiv:2506.02867.

Qwen, :, 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, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv: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. 2025. Qwenlong-l1.5: Post-training recipe for long-context reasoning and memory management. *Preprint*, arXiv:2512.12967.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.

Yaorui Shi, Yuxin Chen, Siyuan Wang, Sihang Li, Hengxing Cai, Qi Gu, Xiang Wang, and An Zhang. 2026. Look back to reason forward: Revisitable memory for long-context llm agents. *Preprint*, arXiv:2509.23040.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. Technical Report MSR-TR-2024-45, Microsoft.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*.

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

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. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *Preprint*, arXiv:2507.02259.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text

embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. 2024. LongRAG: A dual-perspective retrieval-augmented generation paradigm for long-context question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22600–22632, Miami, Florida, USA. Association for Computational Linguistics.

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

A Implementation Details

A.1 Model, Training Data, and Compute

All training runs use Qwen2.5-1.5B-Instruct (Qwen et al., 2025) as the base model and the corresponding Qwen2.5 tokenizer. All compared methods are initialized from the same base checkpoint and are trained under the same chunk-wise memory-agent framework. We implement RL fine-tuning with the veRL framework (Sheng et al., 2025).

We train on RULER-HotpotQA (Yu et al., 2025), a long-document QA dataset constructed by applying the RULER (Hsieh et al.) long-context generation paradigm to HotpotQA (Yang et al., 2018). Each example contains a query, a ground-truth answer, and a long context consisting of 200 documents. The original training split contains 32,768 examples, totaling approximately 973M tokens, and the validation split contains 128 examples, totaling approximately 4.1M tokens. To focus on controlled reward-design evaluation rather than large-scale benchmark optimization, we downsample the training split to 512 examples, corresponding to approximately 15.2M tokens, while keeping the validation split unchanged.

Table 4 summarizes the main training configuration. All main comparison methods use the same data, base model, rollout number, optimization budget, decoding configuration, and compute budget. Each training run requires approximately 440 GPU-hours on 16 NVIDIA H20 GPUs.

ReMemR1 additionally uses method-specific advantage-composition parameters. Since ReMemR1 retrieves previous memories and inserts them back into the recurrent prompt, we set its maximum prompt length to 2048 tokens. These ReMemR1-specific settings are kept consistent with the original ReMemR1 setting and are summarized in Table 5.

Hyperparameter	**Value**
Base model	Qwen2.5-1.5B-Instruct
Tokenizer	Qwen2.5 tokenizer
Training dataset	RULER-HotpotQA
Training examples	512
Validation examples	128
Training tokens	about 15.2M
Validation tokens	about 4.1M
Documents per example	200
Algorithm	GRPO
Rollouts per prompt	8
Rollout temperature	1
Rollout top-p	1
Training batch size	256
PPO mini-batch size	64
PPO epochs	1
Clip epsilon	0.2
Learning rate	1×10^{-6}
Warmup steps	2
KL loss weight	0.001
Training steps	120
Chunk size	5000 tokens
Maximum memory length	1024 tokens
InfoMem coefficient β	0.2
GPU-hours per run	about 440

Table 4: Training configuration used for the main comparison experiments. Decoding, maximum input/output length, and KL-related settings are kept fixed across compared methods.

Parameter	**Value**
Advantage mixing coefficient α	0.8
Action reweighting	false
Maximum prompt length	2048 tokens

Table 5: ReMemR1-specific reward and advantage parameters.

A.2 GRPO and Reward Implementation

Algorithm 1 summarizes the reward-construction procedure used to interface InfoMem with GRPO. The procedure only changes the scalar reward assigned to each rollout; the GRPO optimizer itself is unchanged.

The reward computation is detached from the policy-gradient path. After reward composition, the resulting trajectory-level advantage is assigned to all generated tokens in the rollout, including both memory-update tokens and final-answer tokens. The normalized matching rule used to compute R_{outcome} in Algorithm 1 is used only for training-time rewards; benchmark evaluation uses the task-specific metrics in Section 5.3.1.

Algorithm 1 InfoMem Reward Construction

Input: Query x, long context D, ground-truth answer y^*, rollout group size n
Output: Rollout rewards $\{R_i\}_{i=1}^n$ for GRPO

$\{(M_i, \hat{y}_i)\}_{i=1}^n \leftarrow \textsc{Rollout}(\pi_\theta, x, D, n)$
for $i = 1, \ldots, n$ **do**
  $R_{\text{outcome},i} \leftarrow \mathbb{1}[\text{Norm}(\hat{y}_i) = \text{Norm}(y^*)]$
  $R_i \leftarrow R_{\text{outcome},i}$
end for
$\mathcal{S} \leftarrow \{i : R_{\text{outcome},i} = 1\}$
if $|\mathcal{S}| > 0$ **then**
  for $i \in \mathcal{S}$ **do**
    $r_i \leftarrow r_{\text{gain}}(x, M_i, y^*)$ ▷ teacher-forced scoring
  end for
  if $|\mathcal{S}| = 1$ **then**
    $\tilde{r}_i \leftarrow r_i$ for the only $i \in \mathcal{S}$
  else
    $\mu_{\mathcal{S}} \leftarrow \text{mean}(\{r_i : i \in \mathcal{S}\})$
    $\sigma_{\mathcal{S}} \leftarrow \text{std}(\{r_i : i \in \mathcal{S}\})$
    for $i \in \mathcal{S}$ **do**
      $\tilde{r}_i \leftarrow (r_i - \mu_{\mathcal{S}})/(\sigma_{\mathcal{S}} + \epsilon)$
    end for
  end if
  for $i \in \mathcal{S}$ **do**
    $R_i \leftarrow R_i + \beta \tilde{r}_i$
  end for
end if
return $\{R_i\}_{i=1}^n$

A.3 Training-time Outcome Reward

During training, the sparse outcome reward is computed by normalized boxed-answer matching following the default settings in veRL framework (Sheng et al., 2025). For each rollout, we first keep the last 300 characters of the generated solution and convert them to lowercase. We then extract the answer from the last occurrence of either `\boxed{...}` or `\boxed` in this suffix, as the training prompts explicitly require the model to place the final answer inside a `\boxed{}` expression. If no boxed answer is found, the rollout is assigned zero outcome reward.

The extracted answer is compared against a list of ground-truth answers. A rollout is marked successful if the extracted answer matches any ground-truth answer after normalization:

$$R_{\text{outcome}} = \mathbb{1}\left[\exists y \in \mathcal{Y}^* : \text{Norm}(\hat{y}) = \text{Norm}(y)\right],$$

where $\mathcal{Y}^*$ denotes the set of acceptable ground-truth answers. The normalization follows the string normalization used in the Hendrycks MATH evaluation script from EleutherAI's lm-evaluation-harness (Gao et al., 2024). It removes line breaks, inverse spaces, dollar signs, percentage symbols, `\left`/`\right`, degree markers, and whitespace; normalizes `tfrac`/`dfrac` to `frac`; normalizes escaped backslashes; and canonicalizes decimal forms such as `.5` to `0.5`. The normalized strings are then compared by exact matching.

B Synthetic Hallucinated-Evidence Diagnostic

B.1 Dataset Construction

We construct the hallucinated-evidence diagnostic from SQuAD (Rajpurkar et al., 2016). Each original example contains a question, a supporting context, and a ground-truth answer. To reduce the influence of parametric knowledge, we retain only examples for which the scoring model fails to answer correctly without context but succeeds when conditioned on the original supporting context. This filtering makes the diagnostic focus on context-dependent evidence use rather than memorized answer recall.

For each retained QA pair, we build a three-context candidate set consisting of the original supporting context and two synthetic hallucinated contexts. The hallucinated contexts are generated with Gemini 3 Flash preview (Google, 2026). They are required to preserve the topic, discourse structure, and surface style of the original passage while replacing answer-critical facts so that the resulting text no longer supports the ground-truth answer. This creates a controlled setting in which lexical and topical similarity alone is insufficient for identifying the truly answer-supporting context. After construction, each diagnostic instance contains one positive context and two hallucinated negative contexts, and each evidence score is evaluated by ranking these three candidates for the same question-answer pair.

B.2 Hallucinated Context Generation Prompt

We use Gemini 3 Flash preview to generate two hallucinated contexts for each retained QA pair. The prompt provides the question, the ground-truth answer, and the original supporting context. The generation is constrained to keep the passage topically and stylistically similar to the original context while corrupting answer-critical facts, so that the generated contexts remain plausible distractors but no longer support the ground-truth answer. The prompt template is shown below.

```
You are given a QA sample.
Generate two hallucinated contexts.

Requirements:
1. Keep each context relevant to the question.
2. Preserve the topic, style, and structure of
   the reference context as much as possible.
```

```
3. Modify key supporting facts so that the
   context does not support the gold answer.
4. Do not change the question.
5. Do not remove all relevant information.
6. Do not reveal the gold answer directly.
7. Do not add meta text, warnings, or labels
   indicating that the context is false.

Question:
{question}

Gold answer:
{gold_answer}

Reference context:
{context}

Return format:
{
  "hallucinated_context_1": "...",
  "hallucinated_context_2": "..."
}
```

We conduct manual spot checks to verify the quality of the generated hallucinated contexts, ensuring that they remain fluent and plausible while no longer supporting the ground-truth answer.

B.3 Evidence Score Computation

B.3.1 Information-gain Score

For each candidate context c, we compute the answer-conditioned information-gain score with the same Qwen2.5-1.5B-Instruct model used in training:

$$r_{\text{gain}}(x, c, y^*) = \frac{1}{|y^*|} \log P_\theta(y^* \mid x, c) - \frac{1}{|y^*|} \log P_\theta(y^* \mid x, \emptyset). \tag{11}$$

The two likelihood terms are computed under teacher forcing over the ground-truth answer tokens. The null-context term uses the same query x but replaces the candidate context with an empty context. No answer sampling or decoding is used for this score.

B.3.2 Embedding Similarity

For embedding-based scores, we compare each candidate context against a question-answer query representation. Specifically, we encode the candidate context c and the template *"Question:* `{QUESTION}` *Answer:* `{ANSWER}`*"* with the same embedding model, then compute cosine similarity between the two embeddings. We evaluate this procedure with three open-source embedding models: Qwen3-Embedding-0.6B (Zhang et al., 2025), BGE-M3 (Chen et al., 2024), and multilingual-E5-large-instruct (Wang et al., 2024). Scores are computed independently for each embedding model.

B.3.3 Attention-based Scores

For attention-based scores, we use the same Qwen2.5-1.5B-Instruct model and feed the candidate context, question, and ground-truth answer into the model. We extract attention from the final attention layer. Since the layer contains multiple attention heads, we first average attention weights across heads. Let $A_{u,v}$ denote the resulting head-averaged final-layer attention from token u to token v, $\mathcal{Y}$ denote the answer-token positions, and $\mathcal{C}$ denote the candidate-context token positions.

ATTN-MASS measures the average total attention assigned from answer tokens to candidate-context tokens:

$$s_{\text{mass}} = \frac{1}{|\mathcal{Y}|} \sum_{u \in \mathcal{Y}} \sum_{v \in \mathcal{C}} A_{u,v}. \tag{12}$$

ATTN-TOP1 measures the average strongest context-token attention for each answer token:

$$s_{\text{top1}} = \frac{1}{|\mathcal{Y}|} \sum_{u \in \mathcal{Y}} \max_{v \in \mathcal{C}} A_{u,v}. \tag{13}$$

Both attention scores are higher-is-better and are computed without using the generated answer.

B.4 Metrics: MRR and Z-score SNR

For each question, the candidate set contains one true supporting context and two hallucinated contexts. Since all scores are higher-is-better, we sort the three candidates in descending score order and record the rank of the true supporting context. Given N diagnostic examples, mean reciprocal rank is computed as

$$\text{MRR} = \frac{1}{N} \sum_{q=1}^{N} \frac{1}{\text{rank}_{\text{supporting context}}}. \tag{14}$$

MRR measures whether a score ranks the true supporting context ahead of hallucinated contexts.

We also compute a Z-score signal-to-noise ratio (SNR) to measure the stability of the true-versus-hallucinated separation. For each question q, let s_q^+ denote the score of the true context and $s_{q,1}^-, s_{q,2}^-$ denote the two hallucinated-context scores. We first standardize the three scores within the same question:

$$z_{q,j} = \frac{s_{q,j} - \mu_q}{\sigma_q + \epsilon}, \quad j \in \{+, 1, 2\}, \tag{15}$$

where μ_q and σ_q are computed over the three candidate scores for question q, and ϵ is used for numerical stability. We then compute two standardized

margins for each question:

$$\Delta_{q,k} = z_q^+ - z_{q,k}^-, \quad k \in \{1, 2\}. \tag{16}$$

Pooling all $2N$ margins into Δ, the reported SNR is

$$\mathrm{SNR} = \frac{\mathrm{mean}(\Delta)}{\mathrm{std}(\Delta)}. \tag{17}$$

A higher SNR indicates that the true context is separated from hallucinated contexts with a larger and more stable standardized margin, making the score more suitable as a reward signal.

C Evaluation Protocols

C.1 Benchmark Summary

Table 6 summarizes the evaluation benchmarks used in Section 5.3. InfoMem and the outcome-only GRPO baseline are evaluated under the same chunk-wise memory-agent framework, with the same memory budget, decoding configuration, benchmark subset, and metric for each benchmark. ReMemR1 is evaluated under its original callback-retrieval framework, described in Appendix C.2, to preserve train-test consistency.

MRCR-8needle (OpenAI, 2025) is particularly challenging in the chunk-wise memory-agent setting because the model cannot directly attend over the full context. Instead, it must preserve multiple sparse targets through compressed sequential memory updates. This setting is difficult for a 1.5B model under an 8-needle retrieval task, so absolute sequence-match scores are low across methods. The comparison remains controlled because all methods use the same model scale, memory length, downsampled subset, and evaluation protocol.

For RULER synthetic QA (Yu et al., 2025), we use the 262,144-token evaluation setting and report F1. CorpusQA (Lu et al., 2026) is evaluated on the 128K-token subset, and LongMemEval (Wu et al.) is evaluated on the 115K-token LongMemEval-S subset. For CorpusQA and LongMemEval, we use LLM-as-judge evaluation with Kimi-K2.6; the judging protocol is described in Appendix C.3. For MRCR-8needle, we use the same fixed 100-example subset for all methods to reduce evaluation cost. All model generations during benchmark evaluation are served with vLLM (Kwon et al., 2023). Each evaluation is run once. Following the original ReMemR1 paper and code, ReMemR1 is evaluated with temperature sampling at $t = 0.7$, whereas all other evaluations use greedy sampling. The evaluation datasets are released under permissive MIT or Apache-2.0 licenses and can be used for academic research.

C.2 ReMemR1 Callback-Retrieval Framework

ReMemR1 augments the standard chunk-wise memory-agent framework with callback retrieval over previously generated memories. At each chunk, the model maintains a current memory and a callback query. The callback query is used to retrieve a relevant historical memory from the memories generated in earlier steps, and the retrieved memory is inserted into the next recurrent prompt together with the current memory, the current chunk, and the original question. The model then generates an updated memory and a new callback query for the next step. At the final step, ReMemR1 similarly uses the accumulated memory state and callback-retrieved historical memory to produce the answer.

Because this callback mechanism changes the inference framework rather than only the reward function, we do not force ReMemR1 into the pure chunk-wise setting used by InfoMem. Instead, we keep the shared settings aligned where applicable, including the base model, training data, chunk size, rollout decoding configuration, and benchmark evaluation protocol, while preserving ReMemR1-specific callback and advantage settings.

C.3 LLM-as-Judge Protocol

CorpusQA and LongMemEval are evaluated with the same binary answer-equivalence judging protocol. For each example, the judge receives the query, the ground-truth answer candidates, and the model answer. It does not receive the method name or any training configuration. The judge returns a binary label, where 1 denotes a correct answer and 0 denotes an incorrect answer. The reported score is the percentage of examples labeled correct.

We use Kimi-K2.6 as the judge model and apply the following prompt:

```
SYSTEM_PROMPT = """You are a strict
answer-equivalence judge.
Return 1 only when the model answer fully
contains one ground-truth candidate with
exactly the same meaning: nothing meaningful
is missing and nothing meaningful is added.
Ignore only semantically empty surface
differences, such as articles like "the",
punctuation, spaces, line breaks, case, LaTeX,
or boxing / markup symbols.
```

Benchmark	Task Type	Samples	Sample Context	Metric	Judge	Subset
MRCR-8needle	NIAH,8 needles	100	4K∼1M tokens	Seq. match	N/A	100 / 800
RULER synthetic QA	Sparse QA	128	262K tokens	F1 score	N/A	262K subsets
CorpusQA	Corpus QA	329	128K tokens	LLM judge	Kimi-K2.6	128K subset
LongMemEval	Dialogue QA	500	115K tokens	LLM judge	Kimi-K2.6	LongMemEval-S

Table 6: Summary of evaluation benchmarks and metrics.

```
If the answer is missing any part of the
ground truth, adds any meaningful content,
or you are unsure, return 0.

First write a brief reason. Then write the
final binary judgment inside
<answer></answer>.
The <answer> tag must contain exactly one
character: 0 or 1."""

USER_TEMPLATE = """Query:
{query}

Ground truth candidates:
{ground_truth_candidates}

Model answer:
{answer}

Does the model answer fully contain any
ground-truth candidate with exactly the same
meaning, allowing only semantically empty
surface characters such as "the", punctuation,
spaces, case differences, or boxing / markup
symbols?

Output format:
Reason: <brief reason>
<answer>0 or 1</answer>"""
```

The final decision is parsed from the binary value inside the `<answer>` tag. This protocol treats paraphrases and minor formatting differences as correct only when the predicted answer preserves the complete meaning of one ground-truth candidate. Answers are marked incorrect if they omit essential information, contradict the ground truth, add meaningful unsupported content, or cannot be confidently judged as equivalent.

C.4 Human Calibration of LLM-as-Judge

We further conduct human calibration to assess the reliability of the LLM judge. We hired an undergraduate annotator and presented each example with the query, the ground-truth answer candidates, and the model answer. The annotator assigned the same binary correctness label as the LLM-as-judge protocol, and we compared the human labels with the LLM-judge outputs. We explicitly informed the annotator that the annotations would be used for anonymized academic research. The annotator was paid at an hourly rate ten times the local minimum wage.

For calibration, we inspect three CorpusQA result files, corresponding to the initial instruct model, the Outcome-only GRPO baseline, and InfoMem, as well as one LongMemEval result file from the Outcome-only GRPO baseline. This verification shows high agreement with human judgments, with 99.5% agreement on CorpusQA and 96.8% agreement on LongMemEval.

C.5 Checkpoint Selection Protocol

For the main comparison experiments, all RL methods are trained for 120 optimization steps under the same training budget. During training, we evaluate checkpoints every 2 optimization steps on the validation split described in Appendix A.1. The final checkpoint reported for each method is selected according to the highest validation accuracy under this fixed validation protocol. If multiple checkpoints obtain the same validation accuracy, we select the earlier checkpoint to favor better generalization.

This selection rule is fixed before benchmark comparison and is applied consistently to all training methods. It prevents selecting checkpoints based on test-set performance, while still accounting for the instability of long-context RL fine-tuning. Diagnostic variants are evaluated with the same validation-based selection principle when benchmark scores are reported.

D Prompt Templates

The memory-update and final-answer templates follow the MemAgent setting (Yu et al., 2025). We use the following template for chunk-wise memory updates:

```
TEMPLATE = """
You are presented with a problem, a section of an
article that may contain the answer to the problem,
and a previous memory. Please read the provided
section carefully and update the memory with the
new information that helps to answer the problem.
Be sure to retain all relevant details from the
previous memory while adding any new, useful
information.
```

```
<problem>
{prompt}
</problem>

<memory>
{memory}
</memory>

<section>
{chunk}
</section>

Updated memory:
"""
```

The final answer is generated with the following template:

```
TEMPLATE_FINAL_BOXED = """
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:
"""
```

The information-gain reward r_{gain} uses the same final-answer template: the ground-truth answer is appended after `Your answer:` and scored by teacher-forced per-token log-likelihood, with either the final memory or an empty memory field.

QueryPMI instead uses the following query-conditioned template:

```
TEMPLATE_QUERY_PMI = """
Based on the previous memory,

<memory>
{memory}
</memory>

we can answer the Query: """
```

E Additional Training Curves and Diagnostics

E.1 Main Training Curves

Figures 5 and 6 report the training and validation curves for the main comparison runs and diagnostic variants. All main comparison methods are trained for 120 optimization steps under the same data, model, rollout number, and validation protocol. Checkpoints are evaluated every 2 optimization steps, and the reported benchmark checkpoint is selected according to the validation protocol described in Appendix C.5.

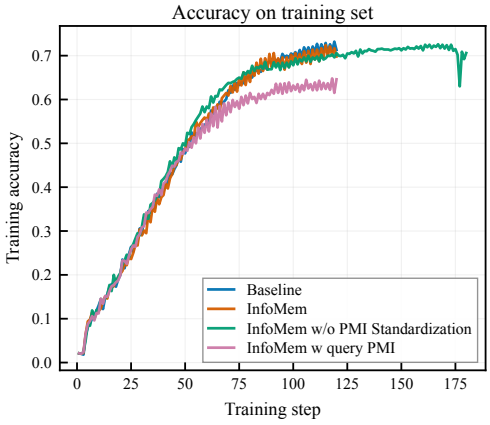


Figure 5: Training curves for the main comparison runs and diagnostic variants.

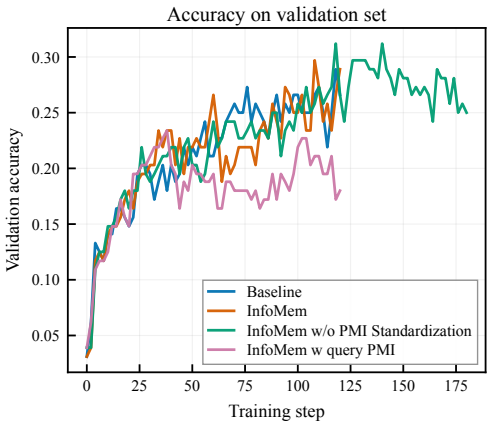
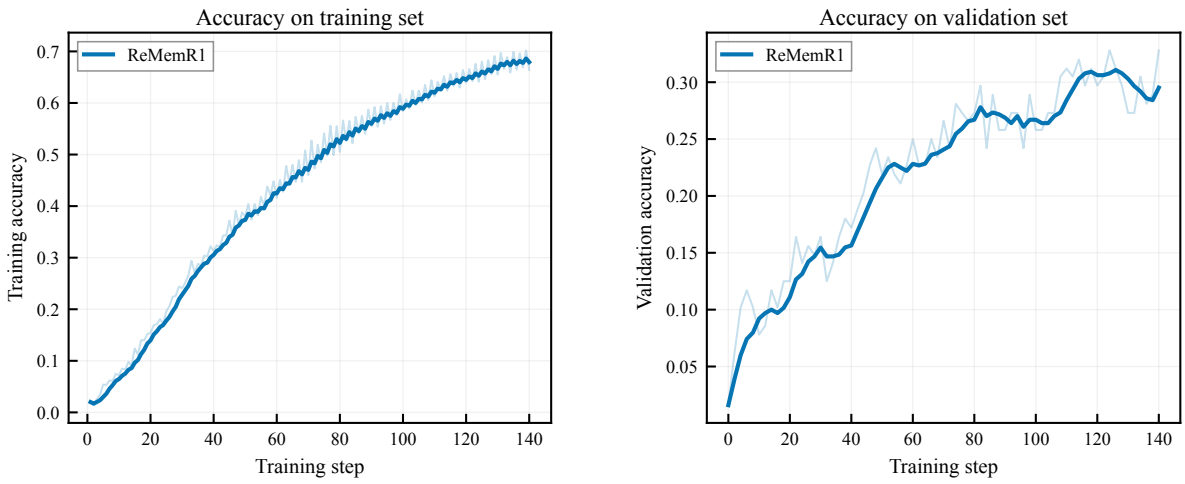


Figure 6: Validation curves for the main comparison runs and diagnostic variants.



(a) Training accuracy.

(b) Validation accuracy.

Figure 7: ReMemR1 training dynamics under its callback-retrieval chunk-wise framework. Both curves are smoothed with a sliding window of 5.

The no-standardization ablation was additionally continued beyond 120 steps, because its curve was still increasing near the end of the original training budget. After continuing the run, the best checkpoint was still step 118, which lies within the original 120-step budget. Therefore, the reported benchmark result for this ablation remains comparable to the other methods under the fixed checkpoint-selection protocol.

ReMemR1 uses a callback-retrieval chunk-wise framework rather than the pure chunk-wise framework used by InfoMem and the outcome-only GRPO baseline. We therefore report its training dynamics separately in Figure 7, instead of overlaying them with the main comparison curves. Because of this framework difference, ReMemR1 is also trained for more than 120 optimization steps.