R^2-MEM: REFLECTIVE EXPERIENCE FOR MEMORY SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Traditional memory systems often organize or compress past information before a query arrives, which may leave out details needed to answer it. Deep memory search addresses this problem by searching full historical records on demand through planning, retrieval, and reflection. However, final outcomes alone do not show which search behaviors to keep or correct. Successful searches may include unnecessary steps, while failed searches may still contain useful ones. We propose **R^2-Mem**, a reflective experience framework for iterative memory search. In the offline stage, a **Rubric-guided Evaluator** assesses planning and reflection steps using search-specific, multi-dimensional rubrics, providing scores, reasons, and practical suggestions. A **self-Reflection Learner** then turns selected high- and low-scoring steps and their feedback into reusable fine-grained reflective experience. During inference, the agent retrieves relevant experience to guide planning and reflection without updating model parameters. Experiments on LoCoMo, HotpotQA, and NarrativeQA show consistent gains in answer quality in agent memory tasks. With Qwen2.5-3B, R^2-Mem achieves a **22.6%** relative F1 gain, while reducing online token usage by **12.9%** and search iterations by **20.2%**. Experiments with **Self-Evo** further show that the backbone can achieve experience-based self-evolution through self-evaluation, without a stronger external evaluator.

1 INTRODUCTION

Long-term memory helps language agents answer questions using past interactions and historical documents. Existing systems organize or compress this information before future questions are known (Xu et al., 2025; Kang et al., 2025; Fang et al., 2026). Although this makes memory easier to access, details left out of a summary may later be needed to answer a question. This motivates **deep search for memory**, which revisits full historical records and builds context on demand (Yan et al., 2025). Through planning, retrieval, and reflection, the agent identifies information needs, gathers evidence, and decides whether further search is required. The quality of these decisions affects both answer quality and search cost.

Historical search trajectories offer useful lessons for improving later searches. Prior work shows that agents can benefit from reusing feedback and experience from past interactions (Shinn et al., 2023; Zhao et al., 2024; Ouyang et al., 2026). For memory search, however, final outcomes alone do not reveal which intermediate behaviors are worth reusing. A successful search may contain redundant retrieval, while a failed search may include useful planning before a later reflection error. Treating every step according to the final outcome can therefore preserve ineffective behavior or discard useful decisions. Learning from these trajectories requires identifying which local behaviors to retain and which to correct.

This calls for clear criteria for judging search steps. Planning must cover the required information, avoid redundant requests, and select suitable tools. Reflection must judge whether the evidence is sufficient and identify what to search for next. These criteria also expose differences within a single step. A plan may cover all information needs yet include unnecessary retrieval, so an overall good-or-bad judgment would lose both practive worth retaining and the behavior worth correcting. Diagnosis alone is insufficient as well. A score does not explain how a future search should proceed, and feedback tied to one question may not apply to another. Userful experience therefore should

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.

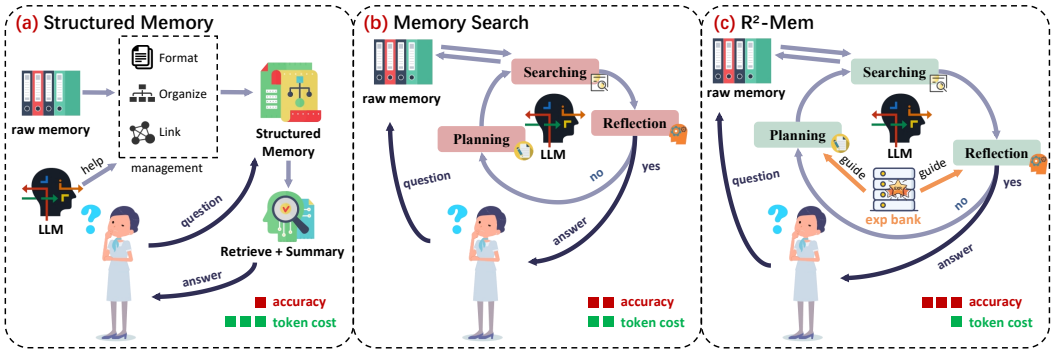


Figure 1: Conceptual comparison of structured memory, iterative memory search, and R^2-Mem. Accuracy and token-cost indicators are schematic, not quantitative results.

capture the specific lession behind an evaluation and make it available when a similar decision arises in a new search.

Towards these needs, we propose **R^2-Mem**, a reflective experience framework for iterative memory search. We design **module-specific, multi-dimensional rubrics** for the different roles of planning and reflection. During offline construction, a **Rubric-guided Evaluator** applies these criteria to historical search steps, providing scores, reasons, and practical suggestions. A **self-Reflection Learner** then turns selected high- and low-scoring steps and their feedback into fine-grained reflective experience. High-scoring steps provide practices to retain, while low-scoring steps provide corrective guidance. These experiences are stored in separate planning and reflection banks. During inference, the agent retrieves relevant experience based on the current query or question–evidence state to guide the corresponding stage. Model parameters and experience banks remain fixed during evaluation. Figure 1 illustrates how this experience guides iterative memory search.

Experiments on LoCoMo, HotpotQA, and NarrativeQA show that R^2-Mem improves answer quality in agent memory tasks. On LoCoMo with Qwen2.5-3B, R^2-Mem achieves a **22.6%** relative F1 gain, while reducing online token usage by **12.9%** and search iterations by **20.2%**. Our **Self-Evo** variant further demonstrates experience-based self-evolution: the backbone learns useful memory-search experience from its own evaluations, without a stronger external evaluator.

2 Related Work

2.1 Iterative Deep Memory Search for LLM Agents

Agent memory systems organize historical information through linked notes, hierarchical stores, and consolidation (Xu et al., 2025; Chhikara et al., 2025; Kang et al., 2025; Fang et al., 2026). Iterative deep memory search further combines lightweight memory with complete historical records, constructing context through planning, retrieval, and reflection (Yan et al., 2025). Access to richer history, however, does not by itself ensure effective search decisions: incomplete plans and inaccurate sufficiency judgments can still cause missed evidence or redundant exploration.

R^2-Mem augments this search process with experience about how to retrieve and assess evidence. By distilling historical planning and reflection behaviors into reusable guidance, it targets repeated search mistakes and redundant exploration without redesigning the underlying memory store.

2.2 Experience Learning and Agent Self-Evolution

Agents reuse verbal reflections, insights, and workflows from past interactions (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2025). ReasoningBank distills reasoning strategies from self-judged successful and failed executions, while ACE incrementally refines contextual playbooks (Ouyang et al., 2026; Zhang et al., 2026b). ReMe further studies fine-grained procedural experience and context-adaptive reuse (Cao et al., 2026); SkillGen and AdaMEM explore step-wise skills and state-dependent guidance (Ding et al., 2026; Zhang et al., 2026c). These studies motivate learning from past

behavior. However, they do not specifically address how to diagnose and reuse individual planning and reflection decisions in iterative agent memory search. This setting produces mixed-quality trajectories: successful searches may contain redundant steps, while failed searches may contain useful partial behaviors.

Within this line of work, R^2-Mem investigates fine-grained reflective experience for agent memory search. It separately diagnoses planning and reflection steps, then abstracts effective practices and corrective lessons from high- and low-scoring behaviors. This step-level design avoids assigning one outcome-level label to an entire trajectory and provides more targeted guidance for later memory search.

2.3 Rubric-based Fine-grained Critique Signals

Rubric-based evaluation makes quality criteria explicit. Rubrics as Rewards turns instance-specific assessments into RL rewards beyond verifiable tasks (Gunjal et al., 2026). Let's Verify Step by Step studies intermediate reasoning supervision, while SAMULE organizes reflection across trajectories, attempts, and tasks (Lightman et al., 2024; Ge et al., 2025). In memory search, an overall score can hide distinct planning and reflection errors and offers no guidance for improvement. Query-specific feedback also requires abstraction before reuse.

R^2-Mem designs separate multi-dimensional rubrics for planning and reflection. Scores select steps for experience construction, while diagnostic reasons and actionable advice guide their abstraction into reusable lessons. This connects evaluation to future search behavior through external experience, enabling adaptation without reinforcement learning or parameter updates.

3 Preliminary

3.1 Deep Memory Search System

A deep memory search system can be described as a cycle of **Planning**, **Searching** and **Reflection**.

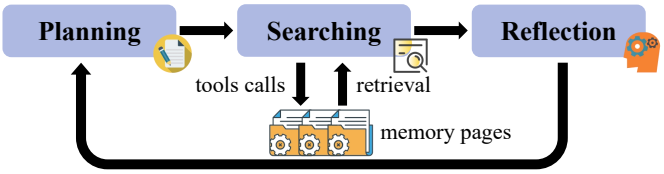


Figure 2: Deep Memory Search System

The overall framework of the memory search system is shown in Figure 2. Formally, the memory search process is formulated as an iterative procedure initialized with the original question Q. Specifically, we initialize the current query as $q_1 = Q$ and the temporary memory as $m_1 = \emptyset$. Let f_{plan}, f_{search}, $f_{\text{integrate}}$, and f_{reflect} denote the planning, searching, integration, and reflection functions, respectively. Each iteration i proceeds as follows.

Planning: Given a current question q_i and current memory m_i, the system first performs structured reasoning to identify explicit information needs. It decomposes the question q_i into concrete information needs $\{q_{i1}, q_{i2}, \ldots, q_{in}\}$ and generates a tool-aware retrieval plan p_i, specifying the retrieval tools (e.g., keyword, semantic, or page-level) and their corresponding inputs. Overall, this process of Planning can be described as:

$$f_{\text{plan}}(q_i, m_i) \rightarrow p_i \tag{1}$$

Searching: Based on the generated plan p_i, the system executes retrieval actions to collect relevant memory from the memory store M by tools. Retrieved results $\{r_{i1}, r_{i2}, \ldots, r_{in}\}$ are aggregated and integrated into a unified intermediate temp memory m_{i+1} that accumulates relevant information:

$$f_{\text{search}}(p_i, M) \rightarrow \{r_{i1}, r_{i2}, \ldots, r_{in}\} \tag{2}$$

$$f_{\text{integrate}}(q_i, m_i, \{r_{i1}, r_{i2}, \ldots, r_{in}\}) \rightarrow m_{i+1} \tag{3}$$

Reflection: After each round of planning and searching, the system evaluates whether the collected temp memory m_{i+1} sufficiently satisfies the original question Q. If critical information is missing,

it formulates a refined follow-up query q_{i+1} and triggers another research iteration. The process continues until sufficient information is collected or the iteration limit is reached.

$$(\text{enough}_i, q_{i+1}) = f_{\text{reflect}}(m_{i+1}, Q), \quad \text{output} = \begin{cases} m_{i+1}, & \text{if enough}_i = \text{true} \\ \text{DeepSearch}(q_{i+1}, m_{i+1}), & \text{otherwise} \end{cases} \tag{4}$$

The resulting memory is then used to answer the original question Q. This closed-loop design enables adaptive information acquisition and allows computation to scale with task complexity.

3.2 Memory Search Trajectory

Given an initial user query Q, the deep memory search process proceeds through multiple iterative interactions with the memory store. To capture this evolving search behavior, we define a *memory search trajectory* as the ordered sequence of intermediate search steps:

$$\tau = (s_1, s_2, \ldots, s_T) \tag{5}$$

where each step s_i is represented as

$$s_i = (q_i, \text{action}_i, m_i), \tag{6}$$

with q_i denoting the current query at iteration i, action_i denoting the corresponding search actions (planning, searching, and reflection), and m_i denoting the current memory state available before executing the actions at iteration i.

The effectiveness of deep search depends on the quality of intermediate actions in the search trajectory. However, evaluating a search trajectory only by its final result cannot reveal the quality of its intermediate planning and reflection decisions. Since incorrect decisions made in early steps may affect later iterations, they can lead to accumulated errors and inefficient search. Therefore, improving deep memory search requires both step-level diagnosis of intermediate decisions and the accumulation of useful search experience along the trajectory.

4 Methodology

4.1 Overview of the Framework

We propose **R^2-Mem**, a reflective experience framework for deep memory search that combines a **Rubric-guided Evaluator** and a **self-Reflection Learner**. Figure 3 illustrates its offline experience construction and online experience-guided search.

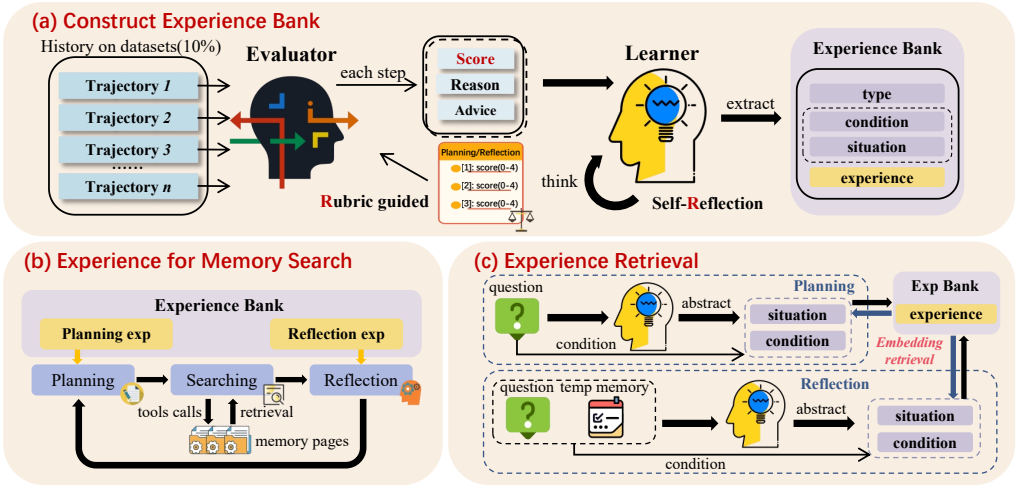


Figure 3: R^2-Mem Framework. (a) Construct an experience bank using 10% of historical trajectories. (b) Perform memory search with experience augmentation on the remaining 90% of datasets. (c) Detailed process of planning and reflection experience retrieval in our framework.

During offline construction, the LLM first generates historical search trajectories. The Evaluator diagnoses their planning and reflection steps using module-specific, multi-dimensional rubrics. The Learner then abstracts selected high- and low-scoring steps into reusable experience, stored in separate planning and reflection banks. During online search, relevant experience is retrieved based on the current query or question–memory state and incorporated into the corresponding search stage. The experience banks remain fixed during evaluation. In Self-Evo, the backbone itself serves as the Evaluator to construct experience from its own trajectories.

4.2 RUBRIC-GUIDED EVALUATING

Let $\tau = (s_1, s_2, \ldots, s_T)$ denote the trajectory generated during the deep search process for question Q, where s_k is the k-th step. The Evaluator provides **step-level diagnosis and scoring**, assessing Planning and Reflection separately using module-specific rubrics (Appendix D.1). For either module, $R = \{d_1, d_2, \ldots, d_n\}$ denotes its rubric dimensions. For each step $s_k \in \tau$, the Evaluator first performs dimension-wise assessment over all rubric dimensions and produces a set of fine-grained scores, which are further aggregated into an overall quality score:

$$\text{Evaluator}_{dim}(s_k, R) \to \{score_k^1, score_k^2, \ldots, score_k^n\}, \quad score_k = \sum_{i=1}^{n} score_k^i \tag{7}$$

where $score_k^i$ is the score under dimension d_i. Based on the aggregated score $score_k$ and the detailed rubric feedback, the Evaluator further generates a corresponding reason r_k and actionable advice a_k. Formally, over the entire trajectory:

$$\text{Evaluator}(\tau, R) \to \{(score_k, r_k, a_k)\}_{k=1}^{T} \tag{8}$$

4.3 SELF-REFLECTION-BASED EXPERIENCE

The Learner abstracts selected steps and their diagnostic feedback into reusable fine-grained reflective experience and stores it in the corresponding experience bank. High-scoring steps provide practices to retain, while low-scoring steps provide corrective guidance. We define the self-reflection process of the Learner as

$$\text{Learner}_{SR}(s_k, r_k, a_k) \to e_q^t, \quad q \in \{\text{good}, \text{bad}\}, \quad t \in \{P, R\} \tag{9}$$

where Learner_{SR} denotes the self-reflection mechanism of the Learner, and e denotes the distilled experience. The label q describes the quality of the source step and is determined by the evaluation score $score_k$: $q = \text{good}$ if $score_k \geq K_{\text{high}}$, and $q = \text{bad}$ if $score_k \leq K_{\text{low}}$. The type label t indicates the functional role of the experience, with $t = P$ for planning-oriented experience and $t = R$ for reflection-oriented experience. Thus, each stored experience is indexed by both quality and function.

4.4 DEEP SEARCH WITH EXPERIENCE

Experience bank. We maintain two experience banks: a *Planning Experience bank* $\mathcal{D}^P$ and a *Reflection Experience bank* $\mathcal{D}^R$. Each bank stores experience pairs of this form:

$$\mathcal{D} = \{(c_j + \sigma_j, e_j)\}_{j=1}^{N}, \tag{10}$$

where c_j denotes the condition, σ_j represents the abstracted situation, "+" denotes string concatenation, and e_j corresponds to the associated experience.

Condition Construction. We denote Q as the overall problem, q_i as the current question, and m_i as the temporary memory at step i. The condition used for experience retrieval depends on the functional role of the experience bank. For planning, the condition is constructed solely from the current query context: $c_i = q_i \quad (Planning)$. For reflection, the condition incorporates both the question Q and the temporary memory: $c_i = [Q + m_{i+1}] \quad (Reflection)$.

Situation Construction. We use LLM_σ to abstract the current condition c_i into a situation σ_i that describes a general information need or evidence state.

$$\text{LLM}_\sigma(c_i) \to \sigma_i \tag{11}$$

Experience Retrieval. Given a condition c_i, we first abstract it into a high-level situation representation σ_i and compute embeddings via $\phi(\cdot)$. We retrieve the top-K most relevant experiences from the experience bank $\mathcal{D}$ by similarity search, where TopK returns the experiences associated with the K highest-scoring entries:

$$\mathcal{E}_i^* = \text{TopK Sim}\big(\phi(c_i + \sigma_i), \phi(c_j + \sigma_j)\big), \quad (c_j + \sigma_j, e_j) \in \mathcal{D} \tag{12}$$

The retrieved experiences are injected into the deep search loop to guide both modules: planning experiences $\mathcal{E}_i^P$ assist question decomposition and tool-aware action generation $f_{\text{plan}}(q_i, m_i, \mathcal{E}_i^P)$, while reflection experiences $\mathcal{E}_i^R$ support sufficiency assessment and follow-up query generation $f_{\text{reflect}}(m_{i+1}, Q, \mathcal{E}_i^R)$.

5 EXPERIMENTS

We evaluate R^2-Mem through the following research questions: **RQ1:** How does R^2-Mem perform on agent memory tasks compared with retrieval, memory management, and deep memory search baselines? **RQ2:** Does R^2-Mem improve answer quality across model sizes while reducing online token usage and search iterations? **RQ3:** How do the Evaluator, Learner, and different experience sources and types contribute to performance? **RQ4:** How sensitive is R^2-Mem to step-selection thresholds and the number of retrieved experiences? **RQ5:** Can the backbone construct useful experience through self-evaluation without a stronger external Evaluator? Further details of the experiments are provided in Appendix B.

5.1 EXPERIMENTAL SETTINGS

Datasets. We evaluate R^2-Mem on three benchmarks: (1) **LoCoMo** (Maharana et al., 2024), a long-term dialogue dataset requiring multi-hop and temporal reasoning; (2) **HotpotQA** (Yang et al., 2018), a multi-hop QA benchmark based on Wikipedia, evaluated with 56K, 224K, and 448K tokens of context; and (3) **NarrativeQA** (Kočiský et al., 2018), which involves reasoning over entire books or scripts.

Baselines. We compare R^2-Mem with four groups of baselines. (1) **Memory-free Retrieval**: a vanilla RAG (Lewis et al., 2020) setting that partitions the input into 2,048-token segments and retrieves the top-5 relevant chunks without explicit memory organization. (2) **Structured Memory Management**: A-Mem (Xu et al., 2025), Mem0 (Chhikara et al., 2025), MemoryOS (Kang et al., 2025), and LightMem (Fang et al., 2026), which pre-organize historical information into explicit memory modules for efficient retrieval. (3) **RL-based Memory**: Memory-R1 (Yan et al., 2026), which uses reinforcement learning to optimize memory management and answering. (4) **Deep Search Memory**: GAM (Yan et al., 2025), which performs iterative planning, searching, and reflection over historical records at runtime. All baseline results are reproduced under our experimental setting. More details of the datasets and baseline implementations are provided in Appendix B.1 and B.2.

Implementation Details. We use Qwen2.5 (3B, 7B, 14B) (Bai et al., 2023) and Llama3.1-8B as backbone models, with GPT-4o as the Evaluator unless otherwise stated. BGE-M3 (Chen et al., 2024) is used for dense retrieval. For HotpotQA and NarrativeQA, we use 20% of the sampled questions for experience construction and the remaining 80% for evaluation. Experience banks remain fixed during evaluation. Unless otherwise stated, R^2-Mem and GAM results are averaged over three random seeds. We report F1 and BLEU1 on LoCoMo, and F1 on HotpotQA and NarrativeQA. Inference uses Python 3.12 and vLLM on A40 GPUs.

5.2 OVERALL PERFORMANCE(RQ1)

R^2-Mem uses GPT-4o as the Evaluator for offline experience construction. Table 1 shows that GAM achieves the strongest overall performance among the baselines on both backbones, supporting iterative deep memory search over memory-free retrieval and structured memory management in this setting. R^2-Mem further improves Overall F1 and BLEU1, showing the added value of reusable planning and reflection experience. It also leads on multi-hop, temporal, and single-hop questions, while Memory-R1 performs best on open-domain questions. Temporal F1 gains over GAM are

particularly large: 15.66 points with Qwen2.5-7B and 7.87 with Llama3.1-8B. R^2-Mem also improves F1 on HotpotQA and NarrativeQA across all tested settings (Table 2). We use GAM as the reference system in subsequent analysis.

Table 1: LoCoMo results with two backbones. R^2-Mem achieves the highest Overall F1 and BLEU1

Model	Method	Multi-hop		Temporal		Open domain		Single-hop		Overall	
		F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1	**F1**	**BLEU1**
Qwen2.5-7B	RAG	12.84	11.02	8.65	7.94	8.97	6.55	9.86	7.50	10.10	8.17
	A-MEM	22.59	16.28	17.69	13.19	14.23	12.53	19.96	13.56	19.63	13.91
	Mem0	19.23	14.17	33.26	24.68	17.33	11.58	28.96	22.45	27.38	20.76
	LIGHTMEM	23.45	17.61	32.16	26.32	13.81	10.31	32.92	26.11	29.90	23.67
	MemoryOS	26.72	17.90	25.65	18.19	22.05	16.95	36.15	28.15	31.45	23.59
	Memory R1	36.55	28.71	39.05	26.03	**28.36**	**23.47**	35.64	28.06	36.07	27.49
	GAM	38.09	29.02	31.10	26.57	23.82	20.19	56.03	49.59	45.77	39.42
	R^2-Mem	**41.62**	**32.34**	**46.76**	**40.67**	25.06	20.78	**59.03**	**53.13**	**51.35**	**44.91**
Llama3.1-8B	RAG	13.29	10.66	9.57	8.34	10.58	8.31	12.29	9.57	11.81	9.44
	A-MEM	23.69	15.28	18.57	13.36	17.59	12.23	21.55	16.23	21.09	15.23
	Mem0	20.43	15.76	34.24	24.91	16.29	10.05	29.12	21.64	27.83	20.55
	LIGHTMEM	24.42	18.66	31.03	25.81	13.31	10.61	33.26	29.25	30.01	25.51
	MemoryOS	27.55	18.91	26.22	16.96	23.15	15.47	38.43	30.15	33.05	24.54
	Memory R1	35.36	26.43	41.26	29.04	**29.34**	**24.56**	37.02	31.23	37.13	29.51
	GAM	36.23	27.71	39.66	33.07	21.04	16.37	50.88	44.58	44.15	37.48
	R^2-Mem	**37.75**	**29.28**	**47.53**	**42.14**	27.48	22.96	**52.16**	**45.73**	**47.13**	**40.66**

Table 2: F1 comparison with GAM on NarrativeQA and HotpotQA.

Model	Method	NarrativeQA	HotpotQA		
		F1	56K	224K	448K
Qwen2.5-3B	GAM	20.19	36.54	31.59	28.05
	R^2-Mem	23.33	41.05	33.98	29.94
	Improv.	**+15.55%**	**+12.34%**	**+7.57%**	**+6.74%**
Qwen2.5-7B	GAM	28.56	44.47	44.63	42.45
	R^2-Mem	31.52	51.53	47.70	44.69
	Improv.	**+10.36%**	**+15.88%**	**+6.88%**	**+5.28%**

5.3 Scaling Behavior and Efficiency Analysis (RQ2)

Table 3 compares R^2-Mem with GAM across three Qwen2.5 backbones on LoCoMo. Relative Overall F1 gains are 22.6%, 12.2%, and 4.5% for 3B, 7B, and 14B, respectively. The declining gains suggest that smaller backbones benefit more from external experience guidance, while the improvement at 14B shows that this guidance remains useful as model capacity increases.

Table 3: R^2-Mem improves answer quality across Qwen2.5 model sizes while reducing online token usage and search iterations on LoCoMo.

Model	Method	Multi-hop		Temporal		Open Domain		Single-hop		Overall		Tokens(M)	Iterations
		F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1		
Qwen2.5-3B	GAM	23.90	17.80	34.78	29.62	17.53	13.67	35.16	31.20	32.00	27.42	44.87	2.47
	R^2-Mem	**30.08**	**23.61**	**37.44**	**32.78**	**20.06**	**16.65**	**44.96**	**39.82**	**39.24**	**34.06**	39.07	1.97
	Gain	+25.9%	+32.6%	+7.6%	+10.7%	+14.4%	+21.8%	+27.9%	+27.6%	+22.6%	+24.2%	-12.9%	-20.2%
Qwen2.5-7B	GAM	38.09	29.02	31.10	26.57	23.82	20.19	56.03	49.59	45.77	39.42	34.01	1.84
	R^2-Mem	**41.62**	**32.34**	**46.76**	**40.67**	**25.06**	**20.78**	**59.03**	**53.13**	**51.35**	**44.91**	28.93	1.52
	Gain	+9.3%	+11.4%	+50.4%	+53.1%	+5.2%	+2.9%	+5.4%	+7.1%	+12.2%	+13.9%	-14.9%	-17.4%
Qwen2.5-14B	GAM	43.89	36.31	49.39	44.14	29.25	24.05	59.83	54.14	52.99	47.09	28.85	1.55
	R^2-Mem	**45.45**	**37.11**	**53.45**	**47.85**	**33.50**	**29.49**	**61.59**	**56.21**	**55.36**	**49.48**	27.73	1.47
	Gain	+3.6%	+2.2%	+8.2%	+8.4%	+14.5%	+22.6%	+2.9%	+3.8%	+4.5%	+5.1%	-3.9%	-5.2%

R^2-Mem also reduces online token usage and search iterations across all three backbones. Token counts in Table 3 are reported in millions (M) and include online inference only. Fewer iterations alone could reflect premature stopping; the accompanying gains in answer quality show that these savings do not come at the expense of answer quality. Thus, experience guidance improves the quality–cost trade-off in memory search. Offline construction costs, including Evaluator and Learner usage, are analyzed separately in Appendix C.1.

5.4 Ablation Study (RQ3)

Table 4 shows that removing either the Evaluator or Learner lowers answer quality across all four categories. Without the Evaluator, experience is summarized from entire trajectories, providing a ReasoningBank-style trajectory-level experience baseline (Ouyang et al., 2026); without the Learner, step-level feedback is reused directly. These comparisons show that the gains come not only from experience reuse, but also from identifying which step-level behaviors to learn from and abstracting their feedback into reusable guidance. Low-scoring steps outperform high-scoring steps alone, highlighting the value of corrective guidance. Combining both improves results, suggesting that effective practices complement corrective lessons. The low-scoring-only variant uses slightly more iterations than the high-scoring-only variant, so its quality advantage is not simply due to earlier stopping. Planning and reflection guidance contribute jointly: the full model achieves the best F1 in three of four categories and the lowest token usage and iteration count. Further analysis is provided in Appendix C.2.

Table 4: Ablation study on rubric-guided components. "w/o" denotes the removal of the module.

Setting	Multi-hop		Temporal		Open Domain		Single-hop		Tokens	Iterations
	F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1		
w/o Evaluator	37.47	28.31	35.12	27.84	23.88	20.25	55.74	48.36	32.41	1.65
w/o Learner	36.26	29.02	32.69	25.51	22.14	19.19	56.36	49.28	35.01	1.88
only high-quality	37.36	28.84	45.59	39.50	23.15	19.77	57.29	51.49	29.11	1.55
only low-quality	39.33	30.27	46.63	40.39	24.35	20.21	59.00	53.04	28.98	1.57
w/o Planning	37.96	29.61	45.61	39.04	24.78	20.63	58.92	52.87	28.99	1.54
w/o Reflection	40.00	31.18	46.02	39.44	**26.03**	20.58	58.07	52.24	29.44	1.56
R^2-Mem (Full)	**41.62**	**32.34**	**46.76**	**40.67**	25.06	**20.78**	**59.03**	**53.13**	**28.93**	**1.52**

5.5 Hyperparameter Analysis(RQ4)

Rubric Threshold Selection for Step Quality. We analyze the effect of step-selection thresholds by varying $(K_{\text{low}}, K_{\text{high}})$, as shown in Figure 4. Steps scoring at or below K_{low} provide corrective guidance, while those scoring at or above K_{high} provide effective practices.

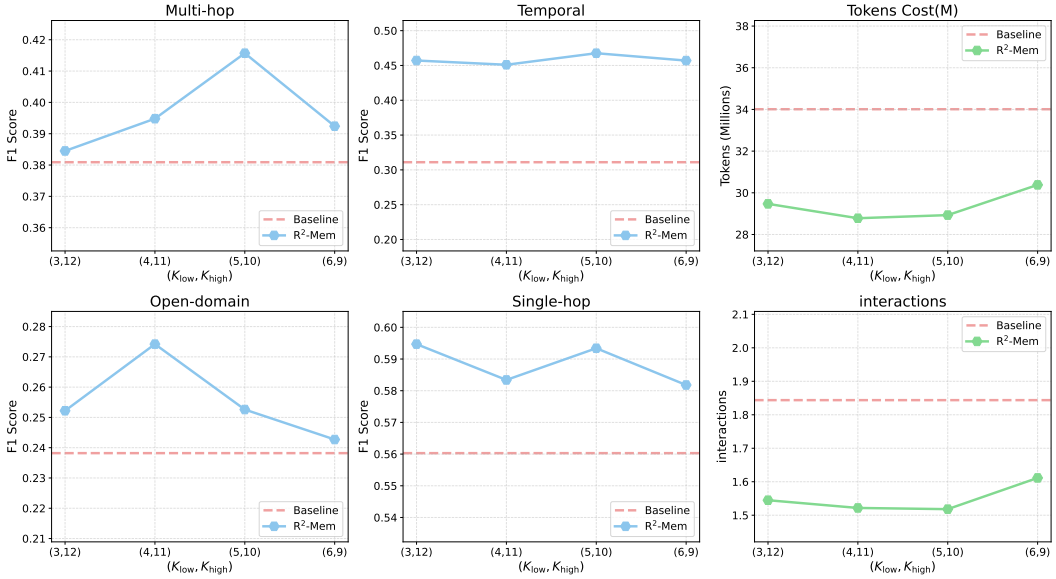


Figure 4: R^2-Mem remains consistently better than the baseline across all settings.

R^2-Mem remains stable under moderate threshold changes. The strict setting $(3, 12)$ may provide insufficient experience, while the loose setting $(6, 9)$ may include weaker guidance. This indicates that the method is not highly sensitive to thresholds within a reasonable range.

Experience Retrieval. We further study the impact of the retrieval size k for experience selection, as shown in Figure 5. Results show that performance is stable under moderate values of k. Smaller

values provide limited guidance, while larger values may introduce redundancy and noise. This indicates a natural trade-off between information richness and retrieval quality. Overall, R^2-Mem is not highly sensitive to k, and a small number of retrieved experiences is sufficient to achieve strong performance. Detailed studies are provided in Appendix C.3.

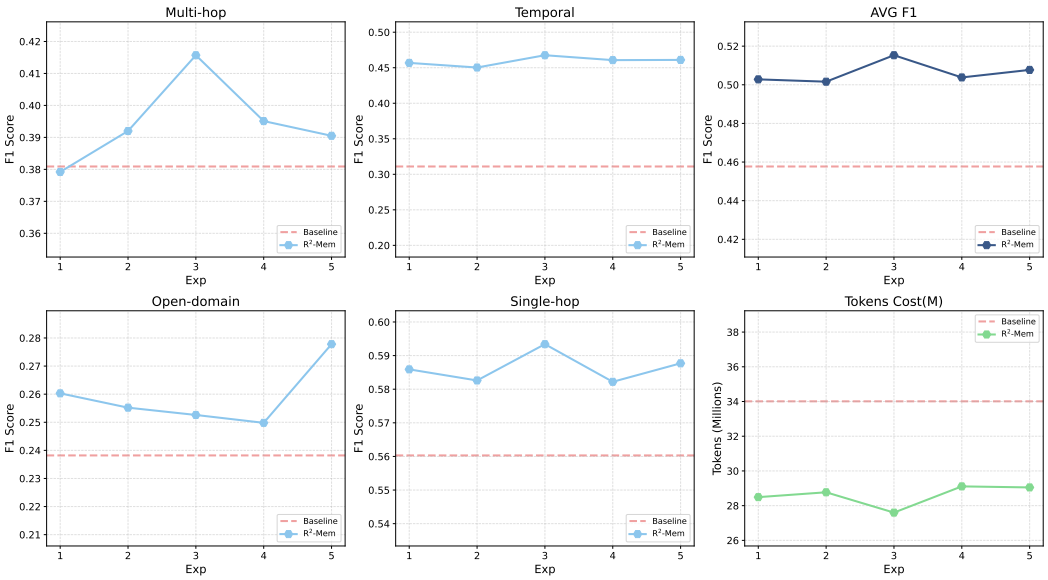


Figure 5: Effect of experience retrieval size k on model performance and retrieval quality.

5.6 Self-Evolution Capability (RQ5)

To test whether the gains depend on GPT-4o, we evaluate **Self-Evo**, which uses the backbone itself instead of GPT-4o as the Evaluator during offline experience construction. Detailed results across Qwen2.5 model sizes are provided in Appendix C.4.

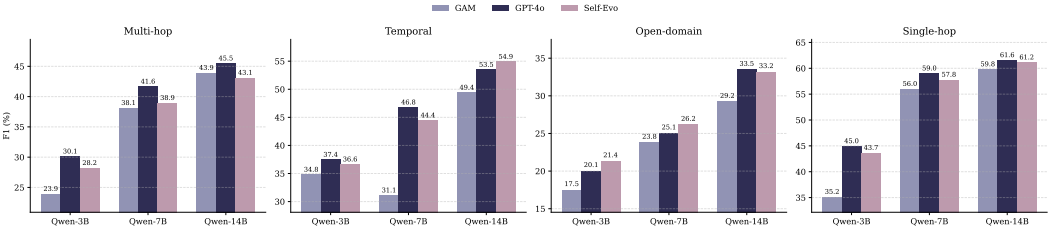


Figure 6: Self-evolution performance of R^2-Mem under different backbone models.

Figure 6 shows that Self-Evo improves Overall F1 over GAM across the tested model sizes on LoCoMo. It remains close to, but below, GPT-4o-based evaluation in Overall F1. These results suggest that stronger feedback helps, but is not required to construct useful experience, supporting experience-based self-evolution through the backbone's own evaluations.

6 Conclusion

We presented R^2-Mem, a reflective experience framework for iterative memory search. Since final outcomes alone cannot reveal which search behaviors to retain or correct, R^2-Mem uses module-specific, multi-dimensional rubrics to diagnose planning and reflection steps, then abstracts their feedback into reusable fine-grained reflective experience. This experience guides later memory search without parameter updates. Experiments show improved answer quality alongside lower online token usage and fewer search iterations. Self-Evo further shows that the backbone can build useful experience through its own evaluations, without a stronger external evaluator. These findings support step-level diagnosis and experience reuse as a practical way to improve agent memory.

AI USE DISCLOSURE

We used generative AI tools to assist with literature search, reading and summarizing related work, and improving the clarity and standardization of the paper's language. All AI-assisted outputs were reviewed and revised by the authors. The authors take full responsibility for the final content of the paper.

ETHICS STATEMENT

This work evaluates language agents on publicly available benchmarks. Our study does not recruit human participants or collect new personal data. Long-term memory agents may nevertheless process sensitive historical information in real-world use. Such systems should obtain appropriate user consent, protect stored data, and support access and deletion controls. Biases in backbone models and evaluators may also affect which information and experience are retained; deployment therefore requires careful auditing and human oversight.

REPRODUCIBILITY STATEMENT

We provide the algorithms, implementation details, prompt templates, hyperparameter analyses, and additional experimental results in the appendix. Our anonymized code and experimental configurations are available at `https://anonymous.4open.science/r/ICLR-2027-R-2-Mem-80E0`.

REFERENCES

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL `https://arxiv.org/abs/2309.16609`.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.829. URL `https://aclanthology.org/2026.findings-acl.829/`.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.137. URL `https://aclanthology.org/2024.findings-acl.137/`.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI*, Frontiers in Artificial Intelligence and Applications, pp. 2993–3000. IOS Press, 2025.

Ruomeng Ding, Wei Cheng, Minglai Shao, and Chen Zhao. SkillGen: Learning domain skills for in-context sequential decision making. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30512–30520, 2026. doi: 10.1609/aaai.v40i36.40305. URL `https://ojs.aaai.org/index.php/AAAI/article/view/40305`.

Pengfei Du. Memory for autonomous llm agents:mechanisms, evaluation, and emerging frontiers, 2026. URL `https://arxiv.org/abs/2603.07670`.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. Lightmem: Lightweight and efficient memory-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dyJ0GWpjJB.

Yubin Ge, Salvatore Romeo, Jason Cai, Monica Sunkara, and Yi Zhang. SAMULE: Self-learning agents enhanced by multi-level reflection. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 16591–16610. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.839. URL https://aclanthology.org/2025.emnlp-main.839/.

Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/cfd7bee7a651ee9af525098ef67a9e45-Abstract-Conference.html.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *EMNLP*, pp. 25961–25970. Association for Computational Linguistics, 2025.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi: 10.1162/tacl_a_00023. URL https://aclanthology.org/Q18-1023/.

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 *NeurIPS*, 2020.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *ACL (1)*, pp. 13851–13870. Association for Computational Linguistics, 2024.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents, 2026. URL https://arxiv.org/abs/2602.01869.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

Jie-Jing Shao, Haiyan Yin, Yueming Lyu, Xingrui Yu, Lan-Zhe Guo, Ivor Tsang, James Kwok, and Yu-Feng Li. Lifting traces to logic: Programmatic skill induction with neuro-symbolic learning for long-horizon agentic tasks. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.01293.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

B. Y. Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General agentic memory via deep research, 2025. URL https://arxiv.org/abs/2511.18423.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2026. URL https://arxiv.org/abs/2508.19828.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pp. 2369–2380. Association for Computational Linguistics, 2018.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=k5nIOvYGCL.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents, 2026a. URL https://arxiv.org/abs/2602.02474.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y. Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://iclr.cc/virtual/2026/poster/10008343.

Yunxiang Zhang, Yiheng Li, Ali Payani, and Lu Wang. AdaMEM: Test-time adaptive memory for language agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026c. URL https://icml.cc/virtual/2026/poster/63989. arXiv:2606.05684.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun Wang, Weiwen Liu, Jianghao Lin, and Weinan Zhang. Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering, 2026. URL https://arxiv.org/abs/2604.08224.

A ALGORITHM

Algorithm 1: Construct Experience Banks

Input: Historical trajectory $\tau = (s_1, s_2, \ldots, s_T)$, module-specific rubrics R^P, R^R, thresholds $K_{\text{high}}, K_{\text{low}}$
Output: Planning and reflection experience banks $\mathcal{D}^P, \mathcal{D}^R$
Let s_k^t denote module $t \in \{P, R\}$ at step k ;
for $k = 1$ **to** T **do**
  foreach $t \in \{P, R\}$ **do**
    $(score_k^t, r_k^t, a_k^t) \leftarrow \text{Evaluator}(s_k^t, R^t)$;
    ; // module-specific evaluation
    if $score_k^t \geq K_{high}$ **then**
      $q \leftarrow$ good ;
    else if $score_k^t \leq K_{low}$ **then**
      $q \leftarrow$ bad ;
    else
      continue ;
    $e_q^t \leftarrow \text{Learner}_{SR}(s_k^t, r_k^t, a_k^t)$;
    ; // experience accumulation
    $c_k^t \leftarrow \text{Cond}(s_k^t),\ \sigma_k^t \leftarrow \text{LLM}_\sigma(c_k^t)$;
    $\mathcal{D}^t \leftarrow \mathcal{D}^t \cup \{(c_k^t + \sigma_k^t, e_q^t)\}$;
    ; // indexed experience storage
return $\mathcal{D}^P, \mathcal{D}^R$;

Algorithm 2: Deep Search with Experience Augmentation

Input: Question Q, memory store M, experience banks $\mathcal{D}^P, \mathcal{D}^R$, maximum iterations $I_{\max}$
Output: Final answer ans
Initialize $q_1 \leftarrow Q$, $m_1 \leftarrow \emptyset$, $i \leftarrow 1$;
while $i \leq I_{\max}$ **do**
  $c_i^P \leftarrow q_i,\ \sigma_i^P \leftarrow \text{LLM}_\sigma(c_i^P)$;
  $\mathcal{E}_i^P \leftarrow \text{TopK Sim}\big(\phi(c_i^P + \sigma_i^P), \phi(c_j + \sigma_j)\big),\ (c_j + \sigma_j, e_j) \in \mathcal{D}^P$;
  ; // planning experience retrieval
  $(\{q_{i1}, \ldots, q_{in}\}, p_i) \leftarrow f_{\text{plan}}(q_i, m_i, \mathcal{E}_i^P)$;
  $R_i \leftarrow f_{\text{search}}(p_i, M)$;
  $m_{i+1} \leftarrow f_{\text{integrate}}(q_i, m_i, R_i)$;
  ; // experience-guided planning and memory update
  $c_i^R \leftarrow [Q + m_{i+1}],\ \sigma_i^R \leftarrow \text{LLM}_\sigma(c_i^R)$;
  $\mathcal{E}_i^R \leftarrow \text{TopK Sim}\big(\phi(c_i^R + \sigma_i^R), \phi(c_j + \sigma_j)\big),\ (c_j + \sigma_j, e_j) \in \mathcal{D}^R$;
  ; // reflection experience retrieval
  $(\text{enough}_i, q_{i+1}) \leftarrow f_{\text{reflect}}(m_{i+1}, Q, \mathcal{E}_i^R)$;
  if $\text{enough}_i = \text{true}$ **then**
    break ;
  ; // experience-guided reflection
  $i \leftarrow i + 1$;
$ans \leftarrow f_{\text{answer}}(Q, m_{i+1})$;
return ans ;

B DETAILED EXPERIMENTAL SETTINGS

Our detailed experimental results and code are available at: `https://anonymous.4open.science/r/ICLR-2027-R-2-Mem-80E0/README.md`.

Implementation details. We use $K_{\text{low}} = 5$, $K_{\text{high}} = 10$, retrieve $k = 3$ experiences, and allow at most three search iterations. The online generator and Learner use a temperature of 0.3, while the GPT-4o Evaluator uses 0.2. For each seed in $\{42, 43, 44\}$, we resample the experience-construction split and rebuild the experience banks before evaluation.

B.1 Details of Datasets

LoCoMo. LoCoMo evaluates long-term memory over multi-session dialogues. Following prior work, we use four task types: *multi-hop retrieval* (category 1), *temporal reasoning* (category 2), *open-domain question answering* (category 3), and *single-hop retrieval* (category 4). We report F1 and BLEU1.

Each dialogue instance in LoCoMo (denoted as Conv-*) is associated with a collection of evaluation questions spanning different categories. Since the category composition varies across dialogues, the difficulty and evaluation emphasis may differ substantially from one conversation to another. To better understand this variation, we analyze the question category distribution of the ten LoCoMo conversations used in our study, as summarized in Table 5.

Table 5: Conversation Questions Distribution

conversation	category 1	category 2	category 3	category 4	sum
Conv-26	32	37	13	70	152
Conv-30	11	26	0	44	81
Conv-41	31	27	8	86	152
Conv-42	37	40	11	111	199
Conv-43	31	26	14	107	178
Conv-44	30	24	7	62	123
Conv-47	20	34	13	83	150
Conv-48	21	42	10	118	191
Conv-49	37	33	13	73	156
Conv-50	32	32	7	87	158
Avg.	28.2	32.1	9.6	84.1	154.0

As shown in Table 5, category composition varies across conversations. For seeds 42, 43, and 44, we select Conv-30, Conv-26, and Conv-47, respectively, for trajectory collection and construct a separate experience bank from each source. R^2-Mem and GAM are evaluated on the same remaining nine conversations in each split, and their results are averaged across the three runs. Conv-26 is used below only for the detailed token-cost analysis.

HotpotQA. HotpotQA is a widely used multi-hop question answering benchmark constructed over the Wikipedia corpus, designed to evaluate a model's ability to perform compositional reasoning across multiple supporting documents. In our experiments, we adopt the long-context memory evaluation setting introduced in MemAgent (Yu et al., 2026), where each question is paired with its gold supporting passages along with a varying number of distracting documents. This setting transforms the original benchmark into a memory-intensive retrieval task, requiring the model to identify relevant evidence under substantial contextual interference.

By varying the number of distracting passages, the MemAgent configuration defines three context-length settings, namely 56K, 224K, and 448K tokens, corresponding to the evaluation subsets `eval_400`, `eval_1600`, and `eval_3200`, respectively.

For each seed in $\{42, 43, 44\}$, we use approximately 20% of the sampled questions for trajectory collection and the remainder for evaluation, rebuilding the experience bank for each run. Tables report the mean over the three runs.

This split separates experience construction from performance evaluation while drawing both sets from the same distribution.

NarrativeQA. NarrativeQA is a long-context question answering benchmark in which each sample provides an entire book or movie script as supporting context, requiring models to retrieve dispersed

evidence and generate answers over extremely long narratives. Compared with document-level QA benchmarks based on shorter passages, NarrativeQA places greater emphasis on sustained context tracking and global memory utilization over tens of thousands of tokens.

For each seed in $\{42, 43, 44\}$, we use 20% of the sampled questions for trajectory collection and the remainder for evaluation, rebuilding the experience bank for each run. Tables report the mean over the three runs.

B.2 Details of Baselines

We provide concise descriptions of all baseline methods for clarity. All baseline results are reproduced by us following the corresponding methods rather than taken from the original papers. We align their backbones, evaluation data, and metrics with our setting. On LoCoMo, GAM is rerun on each of the three splits as described above. Owing to computational cost, the remaining baselines are not rerun across all three splits; preliminary runs showed only minor differences across splits.

RAG (Lewis et al., 2020) performs retrieval-augmented generation by splitting documents into fixed-length chunks (2,048 tokens) and retrieving the top-5 most similar segments via embedding similarity. The retrieved context is directly concatenated with the query for single-pass generation, without explicit memory state or iterative refinement.

Mem0 (Chhikara et al., 2025) maintains an external memory store for multi-session conversations by extracting and consolidating salient information from dialogue history. During inference, it retrieves relevant entries and uses an LLM-based controller to update or prune stored memories.

A-Mem (Xu et al., 2025) organizes interactions into structured memory units inspired by Zettelkasten, where each memory is enriched with attributes such as keywords and contextual descriptions. It constructs a linked memory graph and supports structured retrieval and updates to related entries.

MemoryOS (Kang et al., 2025) models memory as an operating system with short-term, mid-term, and long-term levels. It uses transitions such as FIFO aggregation and page-based consolidation to support long-term storage and retrieval.

GAM (Yan et al., 2025) follows a just-in-time memory search paradigm that separates lightweight offline memory from a full page-store. It uses a dual-module design with a Memorizer for compact memory extraction and a Researcher for online retrieval and integration, enabling dynamic context construction during deep search.

LightMem (Fang et al., 2026) introduces a three-stage memory pipeline inspired by human memory, consisting of sensory filtering, topic-aware short-term aggregation, and sleep-time long-term updates. This design reduces redundancy and inference cost through offline consolidation while preserving long-term recall.

Memory-R1 (Yan et al., 2026) learns memory management policies via reinforcement learning. It adopts a dual-agent design, where a Memory Manager learns structured operations (ADD, UPDATE, DELETE, NOOP) and an Answer Agent retrieves and reasons over memory. Both are optimized using outcome-based RL, enabling adaptive and task-driven memory usage.

C More Detailed Experimental Results

C.1 Amortized Token Cost of Experience Accumulation

A practical concern for R^2-Mem is whether accumulating experience introduces significant additional token overhead, especially when a stronger evaluator (e.g., GPT-4o) is involved. To address this concern, we analyze token consumption during offline experience construction and subsequent conversations, and compare it with GAM. All token counts are reported in millions (M). The main-text efficiency results count online inference only, whereas this appendix also reports the Evaluator and Learner costs incurred during offline experience construction.

For a concrete breakdown, we report the seed-43 run, in which Conv-26 was selected for experience construction. Figure 7 shows token usage across this source conversation and the nine evaluation conversations.

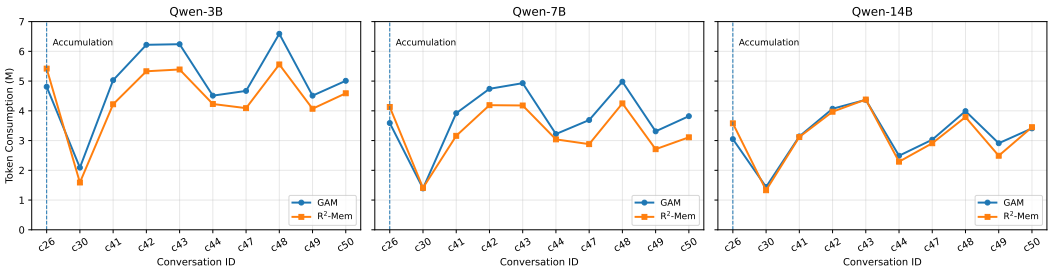


Figure 7: Token consumption in the seed-43 run, where Conv-26 is sampled for experience construction.

Table 6 reports the corresponding token statistics. R^2-Mem uses fewer tokens than GAM on most evaluation conversations across the three backbones.

Table 6: Token consumption (millions) in the representative Conv-26 source run.

Model	**Method**	C26	C30	C41	C42	C43	C44	C47	C48	C49	C50	AVG(9)
Qwen-3B	GAM	4.81	2.09	5.03	6.22	6.24	4.51	4.67	6.59	4.51	5.01	4.986
	R^2-Mem	4.81 + 0.61	1.59	4.22	5.33	5.39	4.23	4.09	5.56	4.07	4.59	4.341
Qwen-7B	GAM	3.59	1.39	3.92	4.74	4.93	3.22	3.69	4.98	3.31	3.82	3.778
	R^2-Mem	3.59 + 0.54	1.41	3.16	4.19	4.18	3.04	2.88	4.25	2.71	3.11	3.214
Qwen-14B	GAM	3.05	1.44	3.14	4.07	4.37	2.49	3.03	3.99	2.91	3.41	3.206
	R^2-Mem	3.05 + 0.53	1.33	3.12	3.97	4.38	2.29	2.91	3.79	2.49	3.45	3.081

The `AVG(9)` columns average the nine evaluation conversations in this representative run; multiplying them by nine gives its online total.

Table 7: Offline construction cost and average online token usage (millions) in the representative Conv-26 source run.

Model	**Offline Construction**			**AVG(9) GAM**	**AVG(9) R^2-Mem**	**Reduction**
	Eval.	**Learn.**	**Total**			
Qwen-3B	0.36	0.25	0.61	4.986	4.341	-12.9%
Qwen-7B	0.29	0.25	0.54	3.778	3.214	-14.9%
Qwen-14B	0.29	0.24	0.53	3.206	3.081	-3.9%

Beyond the shared source-search cost, the Evaluator and Learner add 0.61M, 0.54M, and 0.53M tokens for Qwen-3B, Qwen-7B, and Qwen-14B, respectively. Across the nine evaluation conversations, online token usage is reduced by 5.80M, 5.08M, and 1.12M tokens. Thus, in this representative run, the online savings exceed the additional offline construction cost for all three backbones.

C.2 Detailed Analysis for Ablation Study

Table 4 presents a detailed analysis of different ablation settings in R^2-Mem, focusing on the roles of the Evaluator–Learner framework and different types of experience construction.

(1) Evaluator and Learner. We first study two variants: removing the *Evaluator* and removing the *Learner*. Removing the Evaluator means that the system no longer performs rubric-based step-level evaluation; instead, the Learner directly summarizes experiences from entire historical trajectories without distinguishing fine-grained step quality. In contrast, removing the Learner means that although step-level signals from trajectories are still identified, they are not distilled into abstract reusable experiences, and are instead directly injected as retrieval context during inference.

Results show that removing either component lowers answer quality across all four categories. This supports both identifying which behaviors to learn from and abstracting their diagnostic feedback into reusable guidance.

(2) High-quality vs. low-quality experience. We further compare experience constructed from only high-scoring or only low-scoring steps. The former provides effective practices, while the latter provides corrective guidance.

Experience from low-scoring steps performs better than experience from high-scoring steps alone, highlighting the value of corrective guidance. Combining both further improves results, suggesting that effective practices complement corrective lessons.

(3) Planning and Reflection experience. Finally, we evaluate the effect of removing planning or reflection experience respectively. Removing planning experience eliminates guidance for query decomposition and early-stage search structuring, while removing reflection experience disables feedback signals for evaluating and refining intermediate memory states.

Planning and reflection guidance contribute jointly: the full model achieves the best F1 in three of four categories and the lowest token usage and iteration count.

Overall, these results support step-level diagnosis, experience abstraction, and guidance for both search stages.

C.3 Detailed Analysis for Parameter Sensitivity

We further provide detailed quantitative results to verify the robustness of R^2-Mem under different hyperparameter settings.

Rubric Thresholds. Table 8 shows the results under different step-selection thresholds $(K_{\text{low}}, K_{\text{high}})$. Overall, the model remains stable across all settings, with Overall F1 varying only from 50.17 to 51.35. The best performance is achieved at $(5, 10)$, balancing diagnostic step quality and coverage.

Table 8: Parameter sensitivity analysis of R^2-Mem with different rubric settings.

Rubrics	Multi-hop		Temporal		Open-domain		Single-hop		Overall		Iterations	Tokens(M)
	F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1		
(3,12)	38.45	29.32	45.71	36.38	25.22	20.85	59.47	54.24	50.80	44.08	1.5448	29.47
(4,11)	39.48	30.72	45.10	36.95	27.42	23.32	58.34	52.52	50.37	43.64	1.5217	28.78
(5,10)	41.62	32.34	46.76	40.67	25.06	20.78	59.03	53.13	51.35	44.91	1.5200	28.93
(6,9)	39.24	30.38	45.70	35.25	24.27	19.06	58.18	52.54	50.17	42.99	1.6114	30.38

The strict setting $(3, 12)$ retains only extreme-scoring steps and may provide insufficient experience, whereas the loose setting $(6, 9)$ includes a broader range of steps and may introduce weaker guidance. These results confirm that R^2-Mem is robust to moderate threshold variations and that the default setting is near-optimal.

Experience Retrieval Size. Table 9 reports the results with different numbers of retrieved experiences k. We observe similarly stable performance, with Overall F1 fluctuating within a narrow range of 49.35–51.35.

Table 9: Parameter sensitivity analysis with different numbers of retrieved experiences k.

k	Multi-hop		Temporal		Open-domain		Single-hop		Overall		Iterations	Tokens(M)
	F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1		
1	37.92	28.44	45.68	36.31	26.03	21.56	58.59	52.91	49.51	43.21	1.5357	28.49
2	39.20	30.25	45.03	35.21	25.52	20.95	58.26	52.38	49.55	42.98	1.5354	28.77
3	41.62	32.34	46.76	40.67	25.06	20.78	59.03	53.13	51.35	44.91	1.5200	28.93
4	39.51	29.97	46.07	35.88	24.98	19.24	58.22	52.47	49.35	43.02	1.5419	29.11
5	39.05	30.58	46.10	37.85	27.78	22.72	58.77	52.20	49.69	43.59	1.5224	29.05

Smaller values ($k = 1, 2$) provide limited guidance, while larger values ($k = 4, 5$) may introduce redundant or less relevant experience. The setting $k = 3$ achieves the best overall performance with the fewest iterations, balancing useful guidance and retrieval redundancy.

Overall, the results show that R^2-Mem does not rely on delicate hyperparameter tuning and maintains stable effectiveness and efficiency across a wide range of settings.

C.4 Self-Evolution without GPT-4o Evaluation

Table 10 compares GAM, standard R^2-Mem using GPT-4o as the Evaluator, and Self-Evo using the corresponding backbone itself.

Table 10: Self-Evo results across Qwen2.5 model sizes on LoCoMo.

Model	Method	Multi-hop		Temporal		Open-domain		Single-hop		Overall	
		F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1	F1	BLEU1
Qwen-3B	GAM	23.90	17.80	34.78	29.62	17.53	13.67	35.16	31.20	32.00	27.42
	R^2-Mem	30.08	23.61	37.44	32.78	20.06	16.65	44.96	39.82	39.24	34.06
	Self-Evo	28.18	21.69	36.61	29.86	21.36	16.98	43.67	40.15	38.09	33.32
Qwen-7B	GAM	38.09	29.02	31.10	26.57	23.82	20.19	56.03	49.59	45.77	39.42
	R^2-Mem	41.62	32.34	46.76	40.67	25.06	20.78	59.03	53.13	51.35	44.91
	Self-Evo	38.92	30.22	44.44	38.26	26.20	22.96	57.81	51.66	49.76	43.32
Qwen-14B	GAM	43.89	36.31	49.39	44.14	29.25	24.05	59.83	54.14	52.99	47.09
	R^2-Mem	45.45	37.11	53.45	47.85	33.50	29.49	61.59	56.21	55.36	49.48
	Self-Evo	43.08	34.08	54.88	48.98	33.21	28.67	61.16	56.04	54.93	48.99

Self-Evo improves Overall F1 over GAM at all three model sizes. Its gaps from standard R^2-Mem are 1.15, 1.59, and 0.43 points for Qwen-3B, Qwen-7B, and Qwen-14B, respectively. Stronger external feedback helps, but is not required to construct useful experience.

D Prompt Templates

In this section, we present the specific prompt templates used in our study, including the Rubric-guided Evaluator and the self-Reflection-based Learner for the offline phase, as well as the experience-based memory search for the online phase.

The `REFERENCE_ANSWER` and `MODEL_ANSWER` fields are used only for questions in the offline experience-construction split. Reference answers for evaluation questions are never used to construct experience or during online inference. Self-Evo follows the same data separation.

D.1 Evaluator

Rubric-Guided Evaluator Prompt

```
System Prompt:
You are an expert evaluator for an AI memory deep search
system.
Your task is to assess EACH step in the reasoning TRACE.
Each step contains two modules:  Planning and Reflection.
Planning evaluates whether the retrieval strategy is complete,
non-redundant, tool-aligned, and efficient.
Reflection evaluates whether the system correctly determines
if the current evidence is sufficient, whether it avoids
unnecessary retrieval, and whether the follow-up request
accurately targets missing information.

User Prompt:
QUESTION: {QUESTION}
REFERENCE_ANSWER: {REFERENCE_ANSWER}
MODEL_ANSWER: {MODEL_ANSWER}
TRACE: {TRACE}

For each step in TRACE:
(1) Evaluate the Planning module using the following rubrics:
- Info Needs Coverage (0-3)
```

```
- Info Needs Non-Redundancy (0-3)
- Tool-Info Alignment (0-3)
- Planning Efficiency (0-3)

(2) Evaluate the Reflection module using the following
rubrics:
- Sufficiency Judgment Accuracy (0-3)
- Minimal Sufficiency Recognition (0-3)
- Follow-up Query Quality (0-3)
- Answer Completeness Awareness (0-3)

For each scored module, provide a concise diagnostic reason
and actionable advice for future improvement or maintenance.
The implementation returns them together in one feedback
field; r_k and a_k in the Method section denote its reason and
advice components.
Return ONLY a valid JSON object in the predefined format.
```

JSON Output Format

```
{
  "results": [
    {
      "step": <int number>,
      "module": "Planning",
      "rubrics": {
        "Info Needs Coverage": 0-3,
        "Info Needs Non-Redundancy": 0-3,
        "Tool--Info Alignment": 0-3,
        "Planning Efficiency": 0-3
      },
      "reason and advice": "<diagnostic reason and actionable advice>"
    },
    {
      "step": <int number>,
      "module": "Reflection",
      "rubrics": {
        "Sufficiency Judgment Accuracy": 0-3,
        "Minimal Sufficiency Recognition": 0-3,
        "Follow-up Query Quality": 0-3,
        "Answer Completeness Awareness": 0-3
      },
      "reason and advice": "<diagnostic reason and actionable advice>"
    }
  ]
}
```

D.2 SELF-REFLECTION LEARNER

Planning Experience. The planning prompt abstracts a detailed step into reusable planning strategies. Given a planning step and its detailed information, the LLM is asked to summarize the planning situation and induce a generalized planning experience.

Planning Experience Distillation Prompt

```
System Prompt:
You are an AI TRACE Strategist/Auditor.
Your goal is to derive GENERALIZABLE planning experience from
judged planning traces.
The output experience must follow the form:
IF <abstract situation> THEN <retrieval planning strategy>.

Do not copy concrete surface facts from the trace.
Treat the diagnosed feedback as supervision.
The situation should describe an abstract query type or
information gap.
The experience should describe how to construct:
(1) info_needs, and (2) retrieval tool selection strategy.

User Prompt:
QUESTION: {QUESTION}
JUDGED_PLANNING_TRACE: {TRACE}
DIAGNOSED_FEEDBACK: {REASON_AND_ADVICE}

Your task:
1.  Analyze why this planning trace is high-quality or
low-quality.
2.  Briefly summarize the trace.
3.  Abstract the planning situation without using concrete
entities.
4.  Distill a reusable IF-THEN planning experience that can
guide future retrieval planning.

Return a JSON object with:
{thinking, summary, situation, experience}
```

JSON Output Format

```
{
    "thinking": "<logic analysis grounded in the trace>",
    "summary": "<Briefly describe this Trace(question and plan)>",
    "situation": "<abstract situation>",
    "experience": "IF <abstract situation> THEN <info_need/tool action>"
}
```

Reflection Experience. Reflection prompt abstracts a detailed step into reusable decision-making experience which guides sufficiency judgment and generates a new request.

Reflection Experience Distillation Prompt

```
System Prompt:
You are an AI Memory TRACE Strategist/Auditor.
Your goal is to derive GENERALIZABLE reflection experience
from judged reflection traces.
The output experience must follow the form:
IF <abstract situation> THEN <decision strategy>.

Do not copy concrete facts from the trace.
Treat the diagnosed feedback as supervision.
```

```
The situation should describe an abstract (QUESTION +
temp_memory) state.
The experience should describe how to:
(1) judge whether current memory is sufficient,
(2) decide whether to stop or continue retrieval, and
(3) generate a targeted new_request if needed.

User Prompt:
QUESTION: {QUESTION}
JUDGED_REFLECTION_TRACE: {TRACE}
DIAGNOSED_FEEDBACK: {REASON_AND_ADVICE}

Your task:
1. Analyze why this reflection trace is high-quality or
low-quality.
2. Briefly summarize the trace.
3. Abstract the reflection situation without using concrete
entities.
4. Distill a reusable IF-THEN reflection experience that can
guide future sufficiency judgment.

Return a JSON object with:
{thinking, summary, situation, experience}
```

JSON Output Format

```
{
    "thinking": "<analysis of reflection success/failure>",
    "summary": "<Briefly describe this Trace(question and temp memory)>",
    "situation": "<abstract situation>",
    "experience": "IF <abstract situation> THEN <decision strategy>"
}
```

E Trace and Experience Cases

We first provide a representative memory search trajectory to illustrate the iterative interaction among Planning, temporary memory accumulation, and Reflection-based stopping decisions during online inference.

E.1 Memory Search Trajectory Example

Memory Search Trajectory Example

Question: When did Gina open her online clothing store?

(Step 1)
[Planning]

- Info needs: identify exact opening date of Gina's online clothing store
- Tools: keyword, vector, page_index
- Keyword collection: "Gina online clothing store launch", "Gina opened online clothing store"
- Vector queries: "When did Gina open her online clothing store?"
- Page index: 5, 9

Temp Memory: Gina opened her online clothing store sometime between 29 Jan 2023 and 16 Mar 2023.
[Reflection] "enough = False" → generate new request: "What is the exact date when Gina opened her online clothing store? What did Gina do between 29 January, 2023, and 16 March, 2023?"

(Step 2)
[Planning]

- Info needs: determine exact opening date
- Tools: keyword, vector, page_index
- Keyword collection: Gina, online clothing store, opened, 29 Jan 2023, 16 Mar 2023
- Vector queries: "When did Gina open her online clothing store?", "Find exact launch date of Gina store"
- Page index: 5

Temp Memory: Gina opened her online clothing store on 16 March 2023.
[Reflection] "enough = True", new request = NULL → stop retrieval

Final Memory: Gina opened her online clothing store on 16 March 2023.

E.2 Distilled Experience Cases

We further provide representative entries from the planning and reflection experience bank.

Planning Experience Case

Condition: How long has Caroline had her current group of friends for?

Situation: Time-related query asking for the duration of a relationship or group affiliation.

Summary: This trace asks about the duration of Caroline's current friendship group, but the original plan decomposes it into multiple separate events instead of directly focusing on the required time span. It applies keyword and vector retrieval to collect evidence about when Caroline met her friends and joined the LGBTQ+ community.

Distilled Experience: IF the question asks about the duration of a relationship or group affiliation and involves multiple related events, THEN construct a single info_need centered on the target time frame and use keyword + vector retrieval to locate all relevant dates.

Reflection Experience Case

Condition: Question: When did Melanie run a charity race?
Temp Memory: Melanie ran a charity race for mental health last Saturday.

Situation: A question asks for a specific factual detail, while the current temp_memory only provides a coarse temporal reference rather than the exact target information.

Summary: The question requires the exact date of Melanie's charity race, but the current temp_memory only states that it happened "last Saturday," which is informative but still not sufficient for answering the question precisely.

Distilled Experience: IF the question asks for a specific detail and the temp_memory only contains a general or relative description without that exact detail, THEN set enough = false

and generate a new_request that uses the current coarse clue to retrieve the missing specific information.

F Discussion and Limitations

F.1 Relationship with Skills

Our experience bank is conceptually related to *skills*, since both externalize reusable knowledge beyond model parameters (Du, 2026; Zhou et al., 2026). Prior work represents reusable behavior as workflows, domain skills, or programmatic procedures (Wang et al., 2025; Ding et al., 2026; Shao et al., 2026), while other methods construct and evolve procedural memory from experience (Mi et al., 2026; Cao et al., 2026; Zhang et al., 2026a). In R^2-Mem, each experience is lightweight textual guidance abstracted from a single Planning or Reflection step using module-specific rubric diagnosis. It is retrieved according to the current search state and does not need to form an executable routine. This operational design distinguishes our experience bank from methods that represent reusable knowledge as callable procedures or workflows.

F.2 Broader Impact

This work contributes to the development of more adaptive and reliable long-term memory agents by enabling deep memory search systems to accumulate reusable process-level experience from historical trajectories. Through rubric-guided evaluation and reflective experience reuse, R^2-Mem improves search precision and reduces redundant exploration without updating the backbone parameters. The standard setting incurs a one-time offline cost from GPT-4o evaluation, while Self-Evo reduces reliance on an external evaluator.

The proposed framework may benefit a broad range of memory-intensive applications, including personalized assistants, educational systems, enterprise knowledge agents, and long-context question answering, where stable historical retrieval and process-aware search refinement are essential. Moreover, by introducing explicit evaluator diagnosis and interpretable experience guidance, our method provides a more transparent mechanism for optimizing memory search systems, which may facilitate future research on controllable and self-improving LLM memory systems.

F.3 Limitations

R^2-Mem becomes more useful as it accumulates experience from diverse historical trajectories. At early deployment, a small experience bank may not yet cover enough planning and reflection situations, creating a cold-start issue. Its gains may also be smaller on new types of questions whose information needs or evidence states are not represented in the current bank. Periodically collecting new trajectories and refreshing the bank could improve this coverage.

The current framework constructs experience offline with fixed module-specific rubrics. Although this design provides clear diagnostic criteria, it may not capture every unexpected search error. Future work can study online experience updates, adaptive rubrics, and experience merging to expand coverage while keeping retrieval effective as the bank grows. R^2-Mem is training-free; a further extension could use its step-level rubric signals as process rewards to optimize when to retrieve experience, which experience to select, and how Planning and Reflection use it through reinforcement learning.