The Retriever Should Remember: Experience-Amortized Reranking for Long-Term Agent Memory

Qi Feng Chris Ding Jicong Fan
School of Data Science
The Chinese University of Hong Kong, Shenzhen
qifeng@link.cuhk.edu.cn chrisding@cuhk.edu.cn
fanjicong@cuhk.edu.cn

***Abstract*—Long-term language-model agents accumulate memories across interactions, but their retrievers typically do not accumulate retrieval experience. Semantic retrieval is efficient, but embedding similarity does not always reflect whether a memory contains evidence relevant to the current query. Large language model (LLM) rerankers provide stronger query-conditioned relevance scores, yet stateless reranking repeatedly scores a large candidate pool and discards these scores after each query. We introduce EARM, an experience-amortized reranking framework that treats previously acquired LLM relevance scores as reusable retrieval experience. EARM stores sparse query–memory relevance scores in an online matrix, learns their shared structure through causal matrix completion, and combines a small set of newly observed scores with estimated scores to rerank the remaining candidates. The scoring budget decreases as experience accumulates, changing LLM reranking from a repeated per-query expense into a retrieval capability learned over an agent's lifetime. Experiments on long-term conversational memory show that mixed observed-and-estimated reranking improves answer accuracy over semantic retrieval by up to 6.62% and remains effective when only 17.5% of candidates receive direct LLM relevance scores, thereby substantially reducing the inference overhead of LLM reranking. These results motivate a broader view of agent memory: a long-lived agent should remember not only past content, but also how that content has proved useful for retrieval. Our code is available at https://github.com/FengQi-HITSZ/earm.**

***Index Terms*—agent memory, long-term memory, retrieval, LLM reranking, matrix completion, amortized inference**

I. Introduction

Long-lived language-model agents must carry information across interactions that cannot remain in the immediate context [1], [2]. External memory makes this persistence possible by storing events, user preferences, task trajectories, and other reusable information [3]–[6]. Storage alone, however, does not make an agent remember effectively. As the memory store grows, the system must repeatedly identify a small set of entries that supplies the evidence needed by the current query. Retrieval therefore becomes part of the memory mechanism itself: a stored experience can affect the agent only if it is surfaced at the right time.

Corresponding author

Many current systems, however, exhibit a basic asymmetry: the memory store evolves as interactions accumulate, but the retrieval procedure starts from the same initial state for every query. A typical system retrieves candidates using embedding similarity, optionally uses an LLM to assign a relevance score to each candidate, consumes the selected memories once, and then discards those scores. The same memory may later appear in another candidate set, and a related information need may recur, but the retriever starts over. The agent therefore accumulates *content memory* without accumulating experience about how that content should be accessed. Our starting observation is simple: *agent memory systems are designed to remember past interactions, yet their retrievers forget every retrieval they perform*.

Semantic retrieval alone does not resolve this problem. Dense embedding similarity provides an efficient first-stage filter [7], but representational similarity is not equivalent to query-conditioned memory relevance. A memory can be lexically or topically close to a query while adding no evidence needed for the answer; an apparently distant memory can instead supply a missing temporal, causal, or entity relation. Relevance can also depend on the current dialogue context and on how a memory complements other selected evidence. These conditions explain why a high-similarity candidate pool can still contain substantial noise and why its original order can be a poor answer-oriented ranking [8], [9]. Recent studies reach the same conclusion from complementary perspectives: similarity-based access degrades as relevant evidence becomes temporally distant, and a semantically related memory can still be inappropriate for the current task [10], [11].

LLM rerankers reduce this mismatch by jointly interpreting the query and a candidate memory. Prior work shows that language models can serve as effective text rankers [12]–[14], including under explicit inference budgets [15]. However, conventional reranking treats queries independently. For T queries with N candidates each, pointwise reranking requires up to TN LLM relevance-scoring operations. This cost is especially problematic for long-lived agents because it grows with both the number of interactions and the size of the retrieved pool. More importantly, the cost is repetitive: each score reveals

something about the accessibility of the agent's memory, but a stateless reranker does not preserve that evidence.

We reinterpret these LLM scores as *retrieval experience*. Across an agent's lifetime, the scores form a sparse relevance matrix whose rows are stable memory identities and whose columns are queries. An observed entry records the relevance assigned to one query–memory pair by the LLM reranker. The matrix can exhibit shared structure: some memories are relevant across several information needs; related queries can induce similar relevance patterns; and groups of memories and queries can interact through recurring latent factors. We call this structure a *latent memory accessibility map*. If the map can be learned online, a new query needs only a small number of LLM-scored anchors to locate itself within the map; the system can estimate the relevance of the remaining candidates from accumulated experience.

Based on this view, we introduce EARM (Experience-Amortized Reranking for Memory). For each incoming query, EARM first obtains a semantic candidate pool and asks the LLM to score only a budgeted subset. The subset balances exploitation—sampling candidates from the high-similarity region—with exploration through random sampling from the remaining candidates. EARM inserts these scores into a causal query–memory matrix, updates a biased low-rank model with warm starts, and predicts the unobserved candidate scores. The final ranking uses the observed LLM relevance score when available and the completed score otherwise. Because the matrix also contains memories omitted by the current semantic retriever, completion can further turn that first-stage pool into a soft boundary: historically supported memories outside the pool can be scored and reconsidered when cosine similarity misses them. During cold start, the system observes more scores; as retrieval experience accumulates, a stage-wise schedule reduces the observation budget. The system thus pays an initial cost to learn how its own memory should be retrieved and amortizes this cost over its lifetime.

We evaluate the framework end to end by using sparse LLM relevance scores to rerank retrieved memories before answer generation. Across the evaluated settings, mixed observed-and-completed rankings improve answer accuracy over semantic retrieval by 2.14–6.62% and approach full LLM reranking to within 2.34–2.79% in the strongest completion configurations. The component ablation further shows that completed-score-selected memories add 0.78–2.79% beyond the directly scored memories alone.

This paper makes two contributions:

- We introduce EARM, an experience-amortized reranking framework that maintains LLM relevance scores as persistent retrieval state and combines stratified anchor scoring, causal low-rank completion, and mixed observed–estimated ranking for a sequential query stream.
- We demonstrate that EARM improves answer accuracy over semantic retrieval by up to 6.62%, remains effective at a 17.5% direct-scoring ratio, and obtains consistent additional gains from completed-score-selected memories beyond the observed anchors alone.

II. Related Work

A. Long-Term Agent Memory and Retrieval

Long-term agent memory has evolved from persistent memory streams and tiered storage [1], [3], [4] toward a lifecycle spanning extraction, representation, retrieval, and maintenance [10], [16]. Recent systems extract and consolidate conversational facts or dynamically link memories [5], [6], compress long histories or organize working memory hierarchically [17], [18], structure stores for efficient access [19], [20], learn associative organization [21], or delegate memory operations to smaller models [22]. Long-horizon benchmarks test whether these stores recover multi-hop, temporal, and cross-session evidence [23]–[25]. Retrieval methods therefore enrich memory graphs and propagate activation across them [5], [6], [26], [27], or learn query-conditioned admission beyond embedding similarity [11]. EARM keeps extraction and storage fixed and instead makes query–memory relevance scores persistent across retrievals.

B. Experience-Amortized Reranking

Other agents reuse past trajectories by storing verbal reflections, distilling transferable lessons, or inducing reusable workflows for subsequent tasks [28]–[30]. MemRL also makes retrieval improve from experience by updating memory utilities from environmental feedback and selecting episodic strategies with learned Q-values [31]. EARM differs in both signal and prediction target: it stores pointwise LLM relevance scores for query–memory pairs and predicts unseen pairs through their shared structure, without requiring task rewards or assigning one scalar utility to each memory. LLM rerankers provide this stronger query-conditioned signal by jointly interpreting queries and candidates [14], while budget-constrained rerankers reduce the inference spent on the current query [15]. In contrast, EARM amortizes scoring across a persistent memory store and a sequential query stream. Its bias-aware low-rank completion builds on matrix factorization [32], while its growing-column, fixed-observation-budget setting is related to budgeted column-space recovery [33]. Unlike this prior formulation, the entries are acquired through LLM inference, only causal history is available, and the objective is downstream answer quality rather than matrix reconstruction itself.

III. Problem Formulation

Consider a chronological query stream $q_1, \ldots, q_T$ and a growing memory store $\mathcal{M}_t$ whose entries retain stable identities once written. For query q_t, a first-stage semantic retriever returns a candidate set $C_t \subseteq \mathcal{M}_t$ with $|C_t| = N$. An LLM relevance scorer can then evaluate each query–memory pair:

$$s_{it}^{\mathrm{LLM}} = g_\phi(q_t, m_i) \in [0, 1], \qquad m_i \in C_t, \tag{1}$$

where a higher score indicates that m_i is more relevant to q_t according to the reranker. A full-reranking policy evaluates Eq. (1) for all N candidates independently at every query.

Our setting permits only $B_t < N$ relevance-scoring operations for query q_t. The scorer is applied to an anchor subset

$O_t \subseteq C_t$, with $|O_t| = B_t$. The system must use the observed scores in O_t together with relevance scores accumulated from earlier queries to estimate the unscored candidates $C_t \setminus O_t$ and select K memories for the answer context. The task is therefore sequential sparse reranking: approximate the ranking produced by dense LLM relevance scoring while amortizing scoring effort across the query stream.

The formulation assumes that memory identities remain stable and that query–memory relevance contains sufficient recurring structure for past scores to inform future rankings. Section VIII discusses settings in which these assumptions fail.

IV. Experience-Amortized Memory Reranking

EARM converts sparse LLM relevance scores into a retrieval model that develops over the agent's lifetime. Its five components are semantic candidate generation, experience acquisition, a causal retrieval-experience matrix, online low-rank completion, and mixed reranking. Figure 1 summarizes the complete pipeline from query arrival to answer evaluation.

A. Semantic Candidate Generation

For each query q_t, the semantic retriever computes an embedding similarity s_{it}^{sem} and returns the top N memories. This stage is deliberately high-recall: it narrows the full memory store to a manageable pool but does not determine the final context. Candidate identities are preserved across queries, so a memory retrieved at several times always occupies the same matrix row.

B. Retrieval Experience Matrix

Let $\mathcal{I}_t = \bigcup_{\tau \le t} C_\tau$ denote the set of memory entries that have appeared in at least one semantic candidate set up to query t. We construct a sparse matrix $R^{(t)} \in \mathbb{R}^{|\mathcal{I}_t| \times t}$ with

$$R_{i\tau}^{(t)} = \begin{cases} s_{i\tau}^{\text{LLM}}, & m_i \in O_\tau, \\ \text{unobserved}, & \text{otherwise.} \end{cases} \tag{2}$$

Each column represents a query, each row represents a persistent memory entry, and each observed entry records an LLM relevance score already acquired by the system. An unobserved cell can therefore arise in two ways: the memory was retrieved but not directly scored, or the memory was available in the store but omitted from that query's semantic candidate set. These two cases correspond respectively to the gray and white cells in Figure 1; cells preceding a memory's insertion into a growing store are structurally unavailable and are excluded from completion. We refer to $R^{(t)}$ as the *retrieval-experience matrix* to distinguish it from a temporary per-query score list.

C. Stage-Wise Experience Acquisition

We divide the query stream into experience stages. Let $\ell(t)$ denote the stage containing query t, and let $\rho_{\ell(t)} \in (0, 1]$ be its observation ratio. The relevance-scoring budget is

$$B_t = \left\lfloor \rho_{\ell(t)} N \right\rfloor, \qquad \rho_1 \ge \rho_2 \ge \cdots > 0. \tag{3}$$

The initial stage uses a larger ratio to establish the retrieval-experience matrix. Later stages use progressively smaller ratios as more relevance structure becomes available. Equation (3) describes a predefined maturity schedule rather than an uncertainty-adaptive policy; the instantiated stage boundaries and ratios are experimental hyperparameters.

To construct O_t, we partition C_t into a high-similarity stratum H_t and its complement $C_t \setminus H_t$. Half of the budget is randomly sampled from H_t, and half is randomly sampled from $C_t \setminus H_t$. The former exploits the first-stage retriever by acquiring direct relevance scores in a promising region. The latter explores lower-ranked regions, increases matrix coverage, and prevents the completion model from observing only the semantic retriever's preferred examples. The equal split is fixed across stages; its effect can be isolated through a sampling-policy ablation.

D. Causal Relevance Completion

The retrieval-experience matrix is incomplete by construction: each query column contains direct scores for only its anchor subset. We estimate the missing entries with a bias-aware low-rank model,

$$\widehat{R}_{i\tau} = \mu + \alpha_i + \beta_\tau + p_i^\top z_\tau, \tag{4}$$

where μ is the global relevance level, α_i is a memory-specific bias, β_τ is a query-specific bias, and $p_i, z_\tau \in \mathbb{R}^r$ are latent memory and query factors. The bias terms separate generally high-scoring memories and generally broad queries from pair-specific structure. The interaction $p_i^\top z_\tau$ captures recurring relevance patterns shared across different query–memory pairs.

Let $\Omega_t = \{(i, \tau) : \tau \le t, m_i \in O_\tau\}$ be the entries observed by the time query t is processed. The model minimizes masked reconstruction error over Ω_t together with regularization:

$$\mathcal{L}_t = \sum_{(i,\tau) \in \Omega_t} \left(R_{i\tau} - \widehat{R}_{i\tau} \right)^2 + \mathcal{R}(\alpha, \beta, P, Z). \tag{5}$$

Here, $\mathcal{R}(\alpha, \beta, P, Z) = \lambda\big(\|\alpha\|_2^2 + \|\beta\|_2^2 + \|P\|_F^2 + \|Z\|_F^2\big)$ is an L_2 regularizer on the memory and query biases and their latent factors, with λ controlling the regularization strength. We optimize Eq. (5) with alternating least squares and warm-start each update from the previous solution.

The anchors in the current query column serve a specific role. Historical columns estimate the shared memory parameters α_i and p_i. The observed entries in O_t then constrain the new column parameters β_t and z_t. Once the current query is positioned relative to the historical relevance structure, Eq. (4) estimates scores for $C_t \setminus O_t$. The prediction domain need not stop at C_t. For any memory $m_i \in \mathcal{I}_{t-1} \setminus C_t$ whose row parameters have been learned from earlier observations, Eq. (4) also estimates the corresponding white cell $\widehat{R}_{it}$. Let $E_t \subseteq \mathcal{I}_{t-1} \setminus C_t$ denote such expansion memories and define the ranking domain $D_t = C_t \cup E_t$. Including E_t allows a memory omitted by the current cosine retriever to re-enter the search through its completed relevance score; memories with no historical row support remain cold-start items and require direct acquisition or an inductive initialization. After processing the query, its observed anchors remain in the matrix

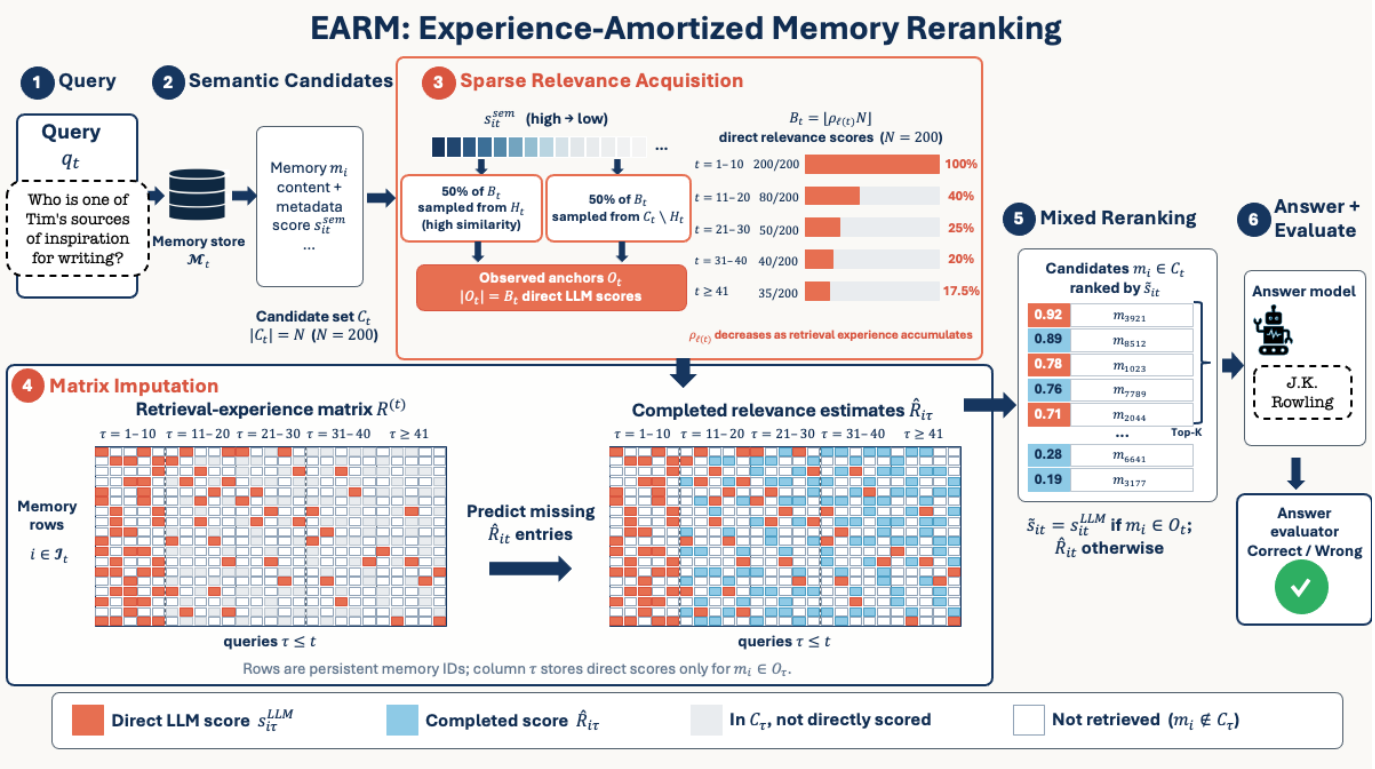

Fig. 1. Overview of EARM. Semantic retrieval forms a candidate set for each query; a budgeted subset receives direct LLM relevance scores; causal matrix completion estimates the missing query–memory scores; and mixed reranking selects the answer context using direct scores when observed and completed scores otherwise. White cells denote memories that were available but not retrieved for a query. Although the experiments rerank within the semantic candidate set, the same completion model can score historically supported white cells and thereby expand the search beyond cosine retrieval. Selected memories are restored to chronological order before answer generation and evaluation.

and refine the shared parameters used by later queries. This repeated update is the mechanism through which reranking effort is amortized across time.

The causal mask is essential. When ranking query q_t, the model may use historical observations and the sampled anchors in column t, but it may not use scores from later queries. The resulting factors represent a causal, evolving map of memory accessibility rather than an offline reconstruction of the complete test matrix.

E. Mixed Reranking and Context Construction

For each memory $m_i \in D_t$, EARM uses

$$\tilde{s}_{it} = \begin{cases} s_{it}^{\mathrm{LLM}}, & m_i \in O_t, \\ \widehat{R}_{it}, & m_i \in D_t \setminus O_t. \end{cases} \tag{6}$$

The experiments use $D_t = C_t$ to evaluate sparse reranking under the same first-stage retrieval pool as the baselines. The expanded form uses $E_t \neq \varnothing$ and turns completion into candidate expansion when cosine retrieval is an unreliable boundary. Memories in D_t are sorted by $\tilde{s}_{it}$, and the highest-scoring K memories form the answer context. Selection order and presentation order serve different purposes: relevance scores determine *which* memories enter the context, after which the selected memories are restored to chronological order to preserve the temporal coherence of the conversation.

This mixed policy retains direct LLM relevance scores where they are available while allowing completed scores to compete for the remaining context slots. As the agent matures, observed scores act as anchors that locate a new query relative to the accessibility map learned from earlier interactions.

V. Experimental Design

We organize the evaluation around three research questions:

1) **RQ1:** How accurately does experience-amortized reranking answer different question types relative to semantic retrieval and full LLM reranking?
2) **RQ2:** Does the advantage over semantic retrieval persist as the direct-scoring ratio decreases across experience stages?
3) **RQ3:** How much of the overall improvement comes from directly scored memories, and how much is added by memories selected through completed scores?

A. Dataset and End-to-End Pipeline

We evaluate on LoCoMo [23], which contains long conversations and questions that test long-term conversational memory. We adopt the memory extraction and storage pipeline of Mem0 [5] and modify only the retrieval and reranking stages. For each conversation, we construct the complete memory store before processing its questions and keep it fixed

throughout evaluation; hence $\mathcal{M}_t = \mathcal{M}$ for every experimental query index t. Only the query columns and observed relevance scores in $R^{(t)}$ accumulate with t—the memory population itself does not grow across experience stages. Questions within each conversation are processed in their dataset order. For every query, the semantic retriever returns a fixed-size candidate set. The current LLM relevance scorer is Qwen3.5-4B-Q8_0 [34], prompted to return a relevance score in $[0, 1]$ for a query–memory pair. After reranking, the top $K \in \{10, 20\}$ memories are ordered chronologically and provided to GPT-4o-mini, which generates the answer. Separately, GPT-4o-mini [35] serves as the LLM-based answer evaluator [36] that compares the generated answer with the reference answer.

B. Implementation Details

The semantic retriever returns $N = 200$ candidates per query. Within each conversation, queries are grouped into blocks of ten. The relevance-scoring budgets are 200, 80, 50, 40, and 35 candidates for query indices 1–10, 11–20, 21–30, 31–40, and 41 onward, corresponding to observation ratios of 100%, 40%, 25%, 20%, and 17.5%. When results are aggregated over the evaluated conversations, the first four stages contain 100 questions each; the final stage contains all remaining questions. After the first block, the sampling policy assigns half of each budget to the high-similarity stratum and half to the remaining candidates. The completion model uses ℓ_2 regularization with coefficient 0.1 and evaluates latent ranks $r \in \{2, 8\}$. ALS runs for at most 5,000 iterations and terminates when the improvement falls below 10^{-6}.

C. Baselines

We compare four retrieval policies:

- **Semantic**: retain the first-stage semantic order and apply no LLM reranker.
- **Full LLM reranking**: score all 200 candidates with the LLM relevance scorer and rank them without completion.
- **Observed-only**: disable score completion and place only directly scored memories among the top K into the answer context.
- **Mixed (EARM)**: rank all candidates using Eq. (6) and return the top K memories.

D. Metrics and Statistical Analysis

The primary end-to-end metric is answer accuracy assigned by the LLM-based answer evaluator. We report overall accuracy and accuracy on multi-hop, open-domain, single-hop, and temporal questions. We count one LLM reranker call for each directly scored query–memory pair. For the component ablation, we report absolute accuracy differences in percent (%) and the average number of directly scored memories in the selected context.

E. Comparison Logic

RQ1 compares sparse completion with the two endpoints of the reranking spectrum: semantic retrieval uses no LLM relevance scores, whereas full LLM reranking scores every candidate. RQ2 groups queries by the experience stages defined above and tests whether completion continues to improve over semantic retrieval when the direct-scoring ratio falls from 100% to 17.5%. RQ3 disables completion while preserving the same observed anchors. The Observed-only–Semantic difference measures the contribution of directly scored memories, while the Mixed–Observed-only difference measures the additional contribution of memories admitted by completed scores. Because Semantic and Mixed both return the same compact context budget $K \in \{10, 20\}$, the end-to-end Semantic–Mixed comparison holds the final context size fixed. These compact budgets also limit the amount of distracting context supplied to the answer model [2], [9], making the ablation primarily sensitive to whether the selected memories contain the evidence needed for the query.

VI. Results

A. Overall and Question-Type Accuracy

Table I compares semantic retrieval, EARM, and full LLM reranking. For Top-10 retrieval, rank-8 EARM improves overall accuracy from 82.21% to 88.83%, a gain of 6.62%, and closes the gap to full LLM reranking to 2.79%. The improvement is largest on multi-hop questions, where accuracy increases by 10.29%, followed by single-hop questions with a 7.25% increase. For Top-20 retrieval, rank-2 EARM performs best among the completion variants, improving overall accuracy by 2.99% and remaining 2.34% below full reranking. Across the evaluation, EARM makes 78,736 LLM reranker calls, compared with 307,982 for full reranking, reducing the total number of calls by 74.43%. With this reduction, the best EARM configuration recovers approximately 70% of the improvement offered by full reranking over Semantic at Top-10 and 56% at Top-20.

Across question types, the gains are most pronounced on multi-hop and single-hop questions, while open-domain and temporal performance remains closer to semantic retrieval. Overall, EARM retains a substantial share of the answer-quality benefit of full reranking while using roughly one quarter of its LLM calls.

B. Accuracy Across Experience Stages

Table II reports accuracy as the observation ratio decreases across query stages. As the direct-scoring ratio falls from 100% to 17.5%, both completion ranks continue to outperform Semantic for both context sizes. In B5+, where only 17.5% of the candidate pairs receive an LLM relevance score, rank-8 EARM improves Top-10 accuracy by 5.53%, while rank-2 EARM improves Top-20 accuracy by 2.46%.

The persistent improvement in B5+ shows that accumulated retrieval experience remains useful when direct scores for the current query are sparse. Thus, EARM maintains a stable advantage over semantic retrieval even after the per-query reranking ratio has fallen to roughly one sixth of the candidate pool.

TABLE I

ANSWER ACCURACY (%) ON LOCOMO BY QUESTION TYPE. FULL LLM RERANKING SCORES ALL 200 CANDIDATES; EARM USES THE STAGE-WISE SPARSE SCORING SCHEDULE. BOLD VALUES ARE THE BEST RESULT FOR EACH TOP-K SETTING.

Top-K	Method	Overall	Multi-hop	Open-domain	Single-hop	Temporal
10	Semantic	82.21	77.30	72.92	85.26	81.31
10	EARM ($r = 2$)	87.27	87.59	73.96	91.44	80.06
10	EARM ($r = 8$)	88.83	87.59	75.00	92.51	84.42
10	Full LLM reranking	**91.62**	**90.43**	**83.33**	**95.24**	**85.67**
20	Semantic	87.08	87.94	78.13	89.18	83.49
20	EARM ($r = 2$)	90.06	90.78	80.21	93.46	83.49
20	EARM ($r = 8$)	89.22	90.43	78.13	92.15	83.80
20	Full LLM reranking	**92.40**	**93.62**	**82.29**	**94.41**	**89.10**

TABLE II

ANSWER ACCURACY (%) ACROSS EXPERIENCE STAGES. B1–B4 CONTAIN 100 AGGREGATED QUESTIONS EACH; B5+ CONTAINS THE REMAINING QUESTIONS. PERCENTAGES IN PARENTHESES ARE THE FRACTIONS OF THE 200 CANDIDATES DIRECTLY SCORED BY THE LLM RELEVANCE SCORER. ANSWER GENERATION AND ANSWER EVALUATION WERE RUN INDEPENDENTLY FOR EACH METHOD, SO SMALL BLOCK-LEVEL FLUCTUATIONS INCLUDE GENERATION AND EVALUATION VARIANCE.

Top-K	Method	B1 (100%)	B2 (40%)	B3 (25%)	B4 (20%)	B5+ (17.5%)	Overall
10	Semantic	76.00	78.00	76.00	77.00	84.12	82.21
10	EARM ($r = 2$)	86.00	86.00	86.00	81.00	88.16	87.27
10	EARM ($r = 8$)	88.00	86.00	86.00	86.00	89.65	88.83
10	Full LLM reranking	85.00	90.00	88.00	88.00	92.98	91.62
20	Semantic	81.00	84.00	89.00	80.00	88.33	87.08
20	EARM ($r = 2$)	88.00	87.00	91.00	86.00	90.79	90.06
20	EARM ($r = 8$)	87.00	91.00	90.00	88.00	89.30	89.22
20	Full LLM reranking	87.00	93.00	94.00	92.00	92.72	92.40

C. Directly Scored versus Completed Memories

Table III separates the two sources of gain in EARM. Observed-only disables matrix completion and retains only memories with direct LLM relevance scores. It improves over semantic retrieval by 1.36–3.83%, demonstrating that the sampled direct scores provide a strong relevance signal. Re-enabling completed scores adds a further 0.78–2.79% in every configuration. The largest completion contribution occurs at $r = 8$, Top-10, where completed-score-selected memories add 2.79% on top of the 3.83% gain provided by directly scored memories.

Together, the three variants provide a direct decomposition of the end-to-end improvement. Semantic and Mixed use the same context budget, establishing the overall benefit of completion-based reranking at fixed K. Observed-only and Mixed use the same directly acquired LLM scores, but only Mixed admits memories selected through imputed scores. The consistently positive 0.78–2.79% Mixed–Observed difference therefore shows that imputation itself improves the selected context, rather than the overall gain arising only from the small set of true LLM scores.

VII. DISCUSSION

A. From Content Memory to Retrieval Memory

Agent-memory research usually asks what an agent should store and how stored content should be represented. Our formulation adds a second persistent object: *retrieval memory*, the accumulated evidence about which memories have been useful for which information needs. Content memory answers "what has the agent experienced?" Retrieval memory answers "what has the agent learned about accessing that experience?" This distinction changes reranking scores from disposable inference traces into state that can improve future retrieval.

B. Why the Method Is More Than a Cache

A cache reuses the result of an identical or near-duplicate query [37]. EARM instead seeks to transfer across distinct queries through shared memory biases, query biases, and latent interactions. It can therefore benefit a new query even when the exact query has never appeared, provided that its sampled anchors connect it to structure learned from previous columns. Candidate overlap helps, but exact repetition is not required.

C. From Candidate Reranking to Search Expansion

The retrieval-experience matrix also makes semantic retrieval a soft rather than irreversible boundary. If a persistent

TABLE III
COMPONENT ABLATION SEPARATING DIRECTLY SCORED AND COMPLETED-SCORE-SELECTED MEMORIES. "OBSERVED IN CONTEXT" IS THE MEAN NUMBER OF SELECTED MEMORIES WITH DIRECT LLM RELEVANCE SCORES. THE OBSERVED-ONLY VARIANT DISABLES COMPLETION; MIXED RE-ENABLES MEMORIES SELECTED BY COMPLETED SCORES.

Rank r	Top-K	Observed in context	Semantic	Observed-only	Mixed (EARM)	Observed − Semantic	Mixed − Observed
2	10	4.59	82.21%	85.84%	87.27%	+3.64%	+1.43%
8	10	5.04	82.21%	86.04%	**88.83%**	+3.83%	**+2.79%**
2	20	7.08	87.08%	88.51%	**90.06%**	+1.43%	+1.56%
8	20	7.49	87.08%	88.44%	89.22%	+1.36%	+0.78%

memory has acquired row parameters from earlier queries but is absent from the current candidate set, its white cell in Figure 1 can still be completed for the new query. Ranking these estimates together with the semantic candidates expands the search domain and allows historically supported memories to enter the answer context even when cosine similarity misses a temporal, causal, or complementary relation. The fixed-pool experiments isolate the reranking effect by setting $D_t = C_t$; the expanded domain $D_t = C_t \cup E_t$ follows from the same completion model without changing the relevance-scoring budget.

D. What Must Be True for Amortization to Work

The main hidden assumption is that query–memory relevance contains persistent cross-query structure. This structure can arise from recurring user interests, memories with stable importance, repeated entities and events, or common types of information need. If every query concerns an unrelated memory subset, if memory identities are unstable, or if the meaning of stored memories changes rapidly, historical scores offer little leverage. The stage-wise results provide evidence that useful structure persists into the lowest-observation stage: with only 17.5% of candidates directly scored, both completion ranks remain more accurate than semantic retrieval.

VIII. LIMITATIONS

The current framework has three limitations. First, its budget schedule is predetermined by query index and does not react to model uncertainty, distribution shift, or cold-start memories. Second, matrix completion assumes reusable low-dimensional structure; highly heterogeneous query streams can violate this assumption. Third, pointwise LLM relevance scores can be noisy, miscalibrated, and prompt-sensitive, so completion can propagate reranker errors rather than only reduce cost.

IX. CONCLUSION

Long-lived agents should not repeatedly forget what their own retrieval decisions have revealed. We presented EARM, a framework that treats LLM relevance scores as retrieval experience, accumulates them in a causal query–memory matrix, and uses a latent accessibility map to rank candidates that were not directly scored. Experiments on LoCoMo show that mixed observed-and-completed rankings improve end-to-end answer accuracy over semantic retrieval, retain their advantage at a 17.5% direct-scoring ratio, and recover additional gains from completed-score-selected memories beyond directly scored anchors alone. More broadly, a useful long-term memory system should not only remember more; it should become better at retrieving what it remembers.

REFERENCES

[1] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3586183.3606763

[2] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024. [Online]. Available: https://aclanthology.org/2024.tacl-1.9/

[3] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, "MemoryBank: Enhancing large language models with long-term memory," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19 724–19 731. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29946

[4] C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, and J. E. Gonzalez, "MemGPT: Towards LLMs as operating systems," 2023. [Online]. Available: https://arxiv.org/abs/2310.08560

[5] P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav, "Mem0: Building production-ready AI agents with scalable long-term memory," 2025. [Online]. Available: https://arxiv.org/abs/2504.19413

[6] W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang, "A-MEM: Agentic memory for LLM agents," in *Advances in Neural Information Processing Systems*, vol. 38, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html

[7] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. [Online]. Available: https://aclanthology.org/2020.emnlp-main.550/

[8] Z. Pan, Q. Wu, H. Jiang, X. Luo, H. Cheng, D. Li, Y. Yang, C.-Y. Lin, H. V. Zhao, L. Qiu, and J. Gao, "SeCom: On memory construction and retrieval for personalized conversational agents," in *International Conference on Learning Representations*, 2025, pp. 91 851–91 885. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2025/hash/e56f394bbd4f0ec81393d767caa5a31b-Abstract-Conference.html

[9] O. Yoran, T. Wolfson, O. Ram, and J. Berant, "Making retrieval-augmented language models robust to irrelevant context," in *International Conference on Learning Representations*, 2024, pp. 29 862–29 883. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/hash/8011b23e1dc3f57e1b6211ccad498919-Abstract-Conference.html

[10] W. Zhou, X. Zhou, S. Han, H. Xu, G. Li, Z. Li, F. Xiong, and F. Wu, "Are we ready for an agent-native memory system?" 2026. [Online]. Available: https://arxiv.org/abs/2606.24775

[11] J. Zhang, K. Chen, J. Ma, Y. Hu, L. He, Y. Zhang, J. Liu, X. Yang, T. Zhang, and R. Jia, "Beyond similarity: Trustworthy memory search for personal AI agents," 2026. [Online]. Available: https://arxiv.org/abs/2606.06054

[12] D. Sachan, M. Lewis, M. Joshi, A. Aghajanyan, W.-t. Yih, J. Pineau, and L. Zettlemoyer, "Improving passage retrieval with zero-shot question generation," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 3781–3797. [Online]. Available: https://aclanthology.org/2022.emnlp-main.249/

[13] W. Sun, L. Yan, X. Ma, S. Wang, P. Ren, Z. Chen, D. Yin, and Z. Ren, "Is ChatGPT good at search? investigating large language models as re-ranking agents," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 14 918–14 937. [Online]. Available: https://aclanthology.org/2023.emnlp-main.923/

[14] Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, L. Yan, J. Shen, T. Liu, J. Liu, D. Metzler, X. Wang, and M. Bendersky, "Large language models are effective text rankers with pairwise ranking prompting," in *Findings of the Association for Computational Linguistics: NAACL 2024*, K. Duh, H. Gomez, and S. Bethard, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 1504–1518. [Online]. Available: https://aclanthology.org/2024.findings-naacl.97/

[15] M. Rashid, J. Meem, Y. Dong, and V. Hristidis, "EcoRank: Budget-constrained text re-ranking using large language models," in *Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 13 049–13 063. [Online]. Available: https://aclanthology.org/2024.findings-acl.773/

[16] Z. Zhang, Q. Dai, X. Bo, C. Ma, R. Li, X. Chen, J. Zhu, Z. Dong, and J.-R. Wen, "A survey on the memory mechanism of large language model-based agents," *ACM Transactions on Information Systems*, vol. 43, no. 6, pp. 1–47, 2025. [Online]. Available: https://doi.org/10.1145/3748302

[17] K.-H. Lee, X. Chen, H. Furuta, J. Canny, and I. Fischer, "A human-inspired reading agent with gist memory of very long contexts," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235. PMLR, 2024, pp. 26 396–26 415. [Online]. Available: https://proceedings.mlr.press/v235/lee24c.html

[18] M. Hu, T. Chen, Q. Chen, Y. Mu, W. Shao, and P. Luo, "HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 32 779–32 798. [Online]. Available: https://aclanthology.org/2025.acl-long.1575/

[19] J. Fang, X. Deng, H. Xu, Z. Jiang, Y. Tang, Z. Xu, S. Deng, Y. Yao, M. Wang, S. Qiao, H. Chen, and N. Zhang, "LightMem: Lightweight and efficient memory-augmented generation," in *The Fourteenth International Conference on Learning Representations*, 2026. [Online]. Available: https://openreview.net/forum?id=dyJ0GWpjJB

[20] J. Liu, Y. Su, P. Xia, S. Han, Z. Zheng, C. Xie, M. Ding, and H. Yao, "SimpleMem: Efficient lifelong memory for LLM agents," 2026. [Online]. Available: https://arxiv.org/abs/2601.02553

[21] J. Zhu, J. Li, C. Zhang, J. Liu, and M. Yang, "HeLa-mem: Hebbian learning and associative memory for LLM agents," in *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, M. Liakata, V. P. Moreira, J. Zhang, and D. Jurgens, Eds. San Diego, California, United States: Association for Computational Linguistics, Jul. 2026, pp. 13 757–13 769. [Online]. Available: https://aclanthology.org/2026.acl-long.625/

[22] J. Zhang, C. Zhang, S. Chen, Z. Huang, P. Zheng, Z. Wang, P. Guo, F. Mo, S.-H. Bae, J. Zou, J. Wei, and Y. Yang, "Lightweight LLM agent memory with small language models," in *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, M. Liakata, V. P. Moreira, J. Zhang, and D. Jurgens, Eds. San Diego, California, United States: Association for Computational Linguistics, Jul. 2026, pp. 12 914–12 929. [Online]. Available: https://aclanthology.org/2026.acl-long.588/

[23] A. Maharana, D.-H. Lee, S. Tulyakov, M. Bansal, F. Barbieri, and Y. Fang, "Evaluating very long-term conversational memory of LLM agents," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 13 851–13 870. [Online]. Available: https://aclanthology.org/2024.acl-long.747/

[24] D. Wu, H. Wang, W. Yu, Y. Zhang, K.-W. Chang, and D. Yu, "LongMemEval: Benchmarking chat assistants on long-term interactive memory," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=UBvm2bIyxz

[25] H. Tan, Z. Zhang, C. Ma, X. Chen, Q. Dai, and Z. Dong, "MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents," in *Findings of the Association for Computational Linguistics: ACL 2025*, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 19 336–19 352. [Online]. Available: https://aclanthology.org/2025.findings-acl.989/

[26] B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su, "HippoRAG: Neurobiologically inspired long-term memory for large language models," in *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 59 532–59 569. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/hash/6ddc001d07ca4f319af96a3024f6dbd1-Abstract-Conference.html

[27] H. Jiang, J. Chen, Y. Pan, L. Chen, W. You, Y. Zhou, R. Zhang, A. Sikora, L. Zhao, Y. Abate, and T. Liu, "SYNAPSE: Empowering LLM agents with episodic-semantic memory via spreading activation," 2026. [Online]. Available: https://arxiv.org/abs/2601.02744

[28] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 8634–8652. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html

[29] A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang, "ExpeL: LLM agents are experiential learners," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, pp. 19 632–19 642, 2024. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/29936

[30] Z. Z. Wang, J. Mao, D. Fried, and G. Neubig, "Agent workflow memory," in *Proceedings of the 42nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 267. PMLR, 2025, pp. 63 897–63 911. [Online]. Available: https://proceedings.mlr.press/v267/wang25bx.html

[31] S. Zhang, J. Wang, R. Zhou, J. Liao, Y. Feng, Z. Li, Y. Zheng, W. Zhang, Y. Wen, Z. Li, F. Xiong, Y. Qi, B. Tang, and M. Wen, "MemRL: Self-evolving agents via runtime reinforcement learning on episodic memory," 2026. [Online]. Available: https://arxiv.org/abs/2601.03192

[32] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[33] C. Kim and M. Bayati, "Recommendation on a budget: Column space recovery from partially observed entries with random or active sampling," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 108. PMLR, 2020, pp. 445–455. [Online]. Available: https://proceedings.mlr.press/v108/kim20a.html

[34] Qwen Team, "Qwen3.5-4B model card," 2026. [Online]. Available: https://huggingface.co/Qwen/Qwen3.5-4B

[35] OpenAI, "GPT-4o mini: Advancing cost-efficient intelligence," Jul. 2024. [Online]. Available: https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[36] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging LLM-as-a-judge with MT-bench and chatbot arena," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 46 595–46 623. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html

[37] F. Bang, "GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings," in *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 212–218. [Online]. Available: https://aclanthology.org/2023.nlposs-1.24/