# G-EVOLVER: LEARNING TO EVOLVE GRAPH MEMORY WITH CONTINUAL STRUCTURAL REFINEMENT

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Memory is central to how large language models (LLMs) sustain long-term, multi-session dialogue. As interactions accumulate, organizing ever-growing memories to support efficient retrieval becomes a critical challenge. Graph structures, which can explicitly model the relationships between memories, have emerged as a promising approach. Existing graph-based memory methods typically extract entity–relation triples from the corpus and link them via shared entities, organizing scattered knowledge into a unified graph. However, such paradigms suffer two fundamental drawbacks. Firstly, relationships based on shared entities fail to capture implicit causal and temporal relationships across sections, resulting in inferior retrieval results. Secondly, queries arrive continuously in real-world deployments while the constructed graph remains static, where structural flaws that are revealed during retrieval cannot be corrected, and rebuilding the entire graph is excessively costly. To this end, we propose **G-Evolver**, which turns the graph from a frozen topology into a structure that evolves continuously with the query stream. Concretely, **G-Evolver** decomposes evolution into two coupled decisions: an extractor first localizes the query-relevant subgraph to be evolved, and an evolver then applies repair operations to it before retrieval; operations verified by retrieval feedback are consolidated into the persistent graph, so that subsequent queries no longer re-encounter the same defects. To train the two modules jointly, we propose Graph Evolution Policy Optimization (GEPO), which turns the verification signal into hierarchical credit over the extraction and the evolution decisions. We further prove that continual evolution converges geometrically to the optimum. Across 4 memory QA benchmarks, **G-Evolver** consistently outperforms 16 competitive baselines[1].

## 1 INTRODUCTION

As Large Language Model (LLM) agents move beyond single-round question answering toward persistent scenarios (Lewis et al., 2020), knowledge from past interactions has to be retained, and access to it becomes decisive for an agent's competence (Chhikara et al., 2025; Pan et al., 2025). Such knowledge accumulates as interaction proceeds: an entity, its later changes, and the events involving it are recorded at different times as separate memory entries. Consequently, when a query draws on such knowledge, the evidence it needs is typically scattered across several entries that must be linked together. Flat vector retrieval scores each entry against the query independently, so entries relevant only through their links to others remain unreachable. Graph structures model such links as edges, allowing retrieval to propagate over multiple hops from query-similar entries to the rest of the evidence. Graph-based memory has therefore emerged as a promising direction (Guo et al., 2024).

Graph-based memory methods (Rasmussen et al., 2025; Edge et al., 2024) typically organize unstructured memory into knowledge graphs and perform retrieval over them. The graph is conventionally built by heuristic rules: HippoRAG (Gutiérrez et al., 2024), for example, extracts entity–relation–entity triples within individual paragraphs and links paragraphs through the entities they share. Recent work (Tsang et al., 2026) makes construction learnable, training the extractor with reinforcement learning so that what it extracts is judged by how well the resulting graph supports retrieval rather than by extraction accuracy alone. In both cases, relations are recovered only within the reach of a single paragraph, while everything across paragraphs is mediated by shared entities. The resulting graph is assembled offline and then used as a fixed structure at retrieval time.

[1] Our code is available at https://anonymous.4open.science/r/G-Evolver-C15E/README.md.



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.

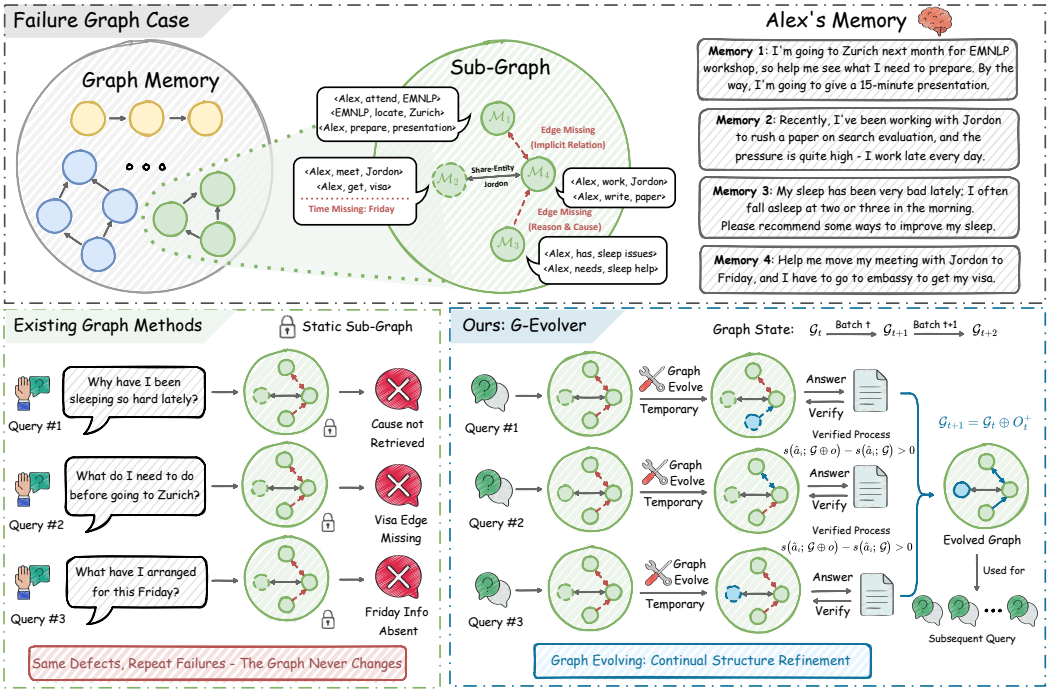



Figure 1: Motivation of **G-Evolver**. **(Top)** Existing graph memory relies solely on shared entities to connect memories across passages, missing relations that hold between memories themselves. **(Left)** These defects are exposed during retrieval yet go uncorrected, so later queries fail in the same place. **(Right)** **G-Evolver** instead evolves the relevant subgraph and consolidates verified edits into graph.

However, these existing paradigms suffer from two fundamental limitations, as illustrated in Figure 1. 🔗 *i) Co-occurrence-based Linking.* Cross-passage edges rest entirely on entity co-occurrence, so a connection is established only where the same entity recurs. Relations that hold between memories rather than between entities, such as causal and temporal dependencies, are consequently absent from the structure. This deficiency is not addressed by learning the triple extractor, whose action space is confined to a single passage and therefore never reaches the cross-passage connections themselves. ❄ *ii) Frozen Topology.* The graph is fixed once constructed. When a query fails and exposes a structural defect, no mechanism exists to correct the graph in response, and the defect persists to undermine every subsequent query that depends on it. Rebuilding the graph from scratch is prohibitively expensive, with a cost that scales linearly with the number of accumulated memories.

To this end, we propose **G-Evolver**, which turns the graph from a frozen topology into a structure that evolves continuously with the query stream. Concretely, **G-Evolver** decomposes evolution into two modules. An extractor first localizes the query-relevant subgraph to be evolved, and an evolver then applies evolution operations to it before retrieval. Operations verified as effective by retrieval feedback are consolidated into the persistent graph, so that subsequent queries no longer re-encounter the same defects. To train the two modules jointly, we propose Graph Evolution Policy Optimization (GEPO), which turns the verification signal into hierarchical credit rather than a single outcome reward: the extractor is credited by whether the subgraph it selects admits effective evolution, while the overall gain is further attributed to each individual operation within the evolver. Besides, we prove that under a repairability condition, continual evolution monotonically improves the expected retrieval score, with its gap to the optimum vanishing geometrically. Our contributions are as follows:

- We identify two structural limitations of existing graph memory and propose **G-Evolver**, a framework that turns the graph into a structure evolving continuously with the query stream: evolution operations take effect provisionally before retrieval and are consolidated once verified by feedback.
- We propose GEPO, which decouples evolution into subgraph extraction and graph evolution, and converts verification signals into hierarchical credit, so that the extractor is credited by the evolvability of the subgraph it selects while individual operations are credited separately.
- Experiments on four long-term memory benchmarks show consistent gains over 16 memory and RAG baselines, and the learned policies transfer to other graph memories without retraining.

# 2 METHOD

## 2.1 PROBLEM SETUP

**Graph Memory Methods.** Given a memory corpus $\mathcal{D} = \{d_1, \ldots, d_N\}$, existing methods construct a memory graph $\mathcal{G}_0 = (\mathcal{V}_0, \mathcal{E}_0)$, where nodes are entities and passage units extracted from $\mathcal{D}$, and edges are induced by entity co-occurrence. Given a query $q$, the retriever links $q$ to seed nodes and propagates relevance over the graph, returning the top-$k$ passages $\mathcal{R}_k(q; \mathcal{G})$, from which a reader model produces the answer $\hat{a}(q; \mathcal{G})$. In existing methods, $\mathcal{G}_0$ remains fixed throughout deployment.

**Evolution Objective.** For a query $q$, let $\mathcal{P}(q; \mathcal{G})$ be the passages retrieved from $\mathcal{G}$ and $\mathcal{P}^\star$ the passages required to answer it. The retrieval score $s\big(\mathcal{P}(q; \mathcal{G}), \mathcal{P}^\star\big) \in [0, 1]$ measures how well the former covers the latter, abbreviated as $s(q; \mathcal{G})$ below. Assume queries are drawn from a distribution $q \sim \mathcal{Q}$. The goal of graph evolution is to reach the topology maximizing the expected score:

$$\mathcal{G}^\star \;=\; \arg\max_{\mathcal{G}} \; \mathbb{E}_{q\sim\mathcal{Q}}\big[\, s\big(\mathcal{P}(q; \mathcal{G}),\, \mathcal{P}^\star\big)\big]\,, \tag{1}$$

where $s(\cdot, \cdot) \in [0, 1]$ measures how well the retrieved passages cover the required ones.

## 2.2 CONTINUAL GRAPH EVOLUTION

A static graph carries structural defects that retrieval continually exposes, and constructing $\mathcal{G}^\star$ is infeasible. Each arriving query samples $\mathcal{Q}$ and reveals where the current structure falls short, so **`G-Evolver`** approaches $\mathcal{G}^\star$ by evolving the graph along the query stream. As shown in Figure 1, each query triggers one evolution episode: a local subgraph is extracted, operations are applied provisionally over multiple rounds, and the effective ones are consolidated into the persistent graph. Both modules are trained with GEPO (Sec. 2.3); here we describe their inference-time behavior.

**Query-Relevant Subgraph Extraction.** Evolving the entire graph for every query is computationally prohibitive and largely redundant, since a single query touches only a small region of $\mathcal{G}_t$. We therefore confine evolution to a query-conditioned local subgraph. Given $q_t$ and the current graph, the extractor $\pi_{\mathsf{extract}}$ selects the passages whose surrounding structure is to be evolved,

$$\mathcal{S}_t \;=\; \pi_{\mathsf{extract}}\big(q_t,\, \mathcal{G}_t\big), \tag{2}$$

and the local view handed to the evolver is

$$\mathcal{G}_t[q_t] \;=\; \big(\mathcal{S}_t,\, \mathcal{T}(\mathcal{S}_t),\, \mathcal{E}(\mathcal{S}_t)\big), \tag{3}$$

where $\mathcal{T}(\mathcal{S}_t)$ collects the triples extracted from each passage in $\mathcal{S}_t$, and $\mathcal{E}(\mathcal{S}_t)$ denotes the edges of $\mathcal{G}_t$ among these passages, encoding how they are currently connected.

**Subgraph Evolution.** Given the local view, the evolver $\pi_{\mathsf{evolve}}$ emits evolution operations, and because the effect of an edit on retrieval is not visible in that view, it edits over at most $R$ rounds rather than committing to a single guess,

$$O_t^{(r)} \;=\; \pi_{\mathsf{evolve}}\big(q_t,\, \mathcal{G}_t[q_t] \oplus O_t^{<r},\, \mathcal{P}(q_t; \mathcal{G}_t \oplus O_t^{<r})\big), \qquad O_t \;=\; \textstyle\bigcup_{r=1}^{R} O_t^{(r)}, \tag{4}$$

where each round applies its operations *provisionally* and re-executes retrieval, so the next round sees the ranking its own edits produced. Three edits are available: inserting a triple stated in a passage but missing from its extraction, linking two related passages with a semantic edge, and rewriting a triple that misrepresents its source. Abstention, $O_t^{(r)} = \emptyset$, serves twice: in the first round it keeps a sound view untouched, and later it ends the episode, so how long to keep editing is itself learned. The final ranking $\mathcal{P}(q_t; \mathcal{G}_t \oplus O_t)$ is what the reader answers from.

**Verification and Consolidation.** Operations remain provisional because committing them blindly would let a single erroneous edit corrupt the shared graph and propagate to all future queries. Verification takes place after $q_t$ has been answered and introduces no latency to the current query. Let $\mathcal{N}(q_t)$ denote the neighboring queries of $q_t$, those whose retrieved passages overlap with $q_t$'s. Operations are validated sequentially: starting from $A = \emptyset$, each $o \in O_t$ is accepted into $A$ if

$$s(q_t; \mathcal{G}_t \oplus A \oplus o) > s(q_t; \mathcal{G}_t \oplus A), \qquad s(q'; \mathcal{G}_t \oplus A \oplus o) \geq s(q'; \mathcal{G}_t) \;\; \forall q' \in \mathcal{N}(q_t), \tag{5}$$

and $O_t^+$ is the final $A$, so that as a whole it strictly improves $q_t$ and lowers no neighbor. Only this validated subset is *consolidated* into the persistent graph, $\mathcal{G}_{t+1} = \mathcal{G}_t \oplus O_t^+$. The graph thus accumulates verified structure toward $\mathcal{G}^\star$ in Eq. (1), with each repair reused by later queries.

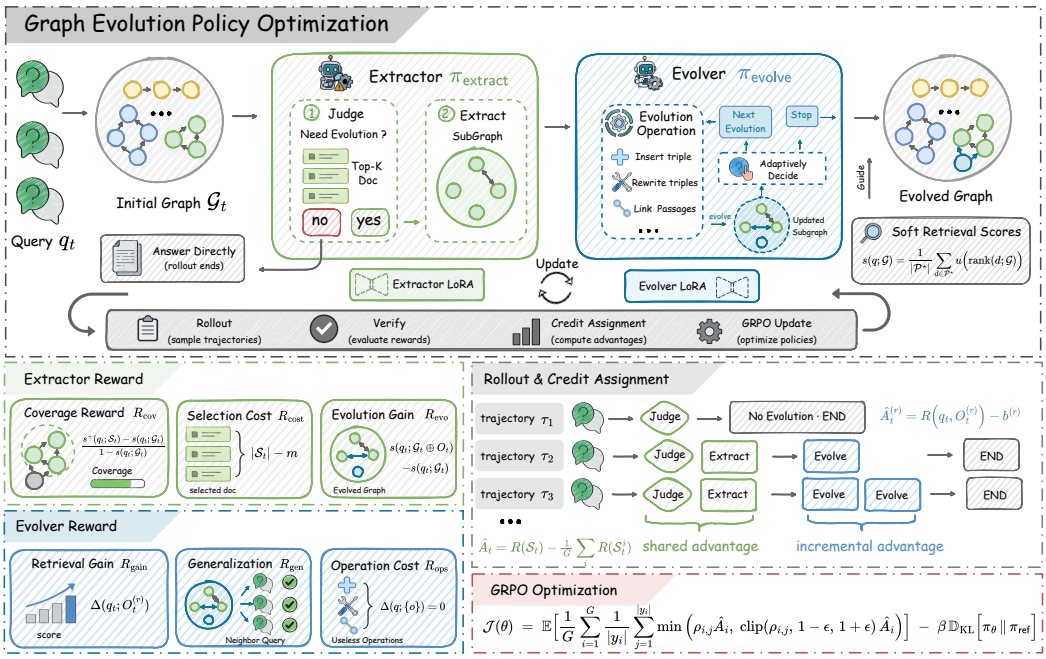



Figure 2: Overview of GEPO. **(Top)** Training process of GEPO, where the extractor and evolver are optimized in alternation. **(Bottom)** Module-specific rewards and hierarchical credit assignment.

### 2.3 GEPO: Graph Evolution Policy Optimization

Learning to evolve the graph is a challenging task. The optimal graph $\mathcal{G}^\star$ is defined only in expectation over $\mathcal{Q}$, and no annotation specifies which edges a passage should carry. The verification step in Sec. 2.2 nevertheless makes the gain of each operation measurable once the query is answered, and GEPO takes this signal as the reward for reinforcement learning. The signal is, however, an aggregate one, reflecting both the region the extractor selected and the operations the evolver applied to it. Assigning it uniformly would penalize a well-chosen subgraph for ineffective edits made within it, and credit a poor one for edits that happened to help. GEPO therefore assigns credit hierarchically, so that each module is optimized against the part of the outcome it is responsible for.

**Retrieval Signal.** A natural reward would be the quality of the answer, but generation is inherently noisy: a multiple-choice question can be answered correctly by chance, and lexical measures such as F1 do not certify that a free-form answer is right. Moreover, what graph evolution controls is not how an answer is generated, but whether a query reaches the passages that genuinely answer it, so we place the reward on retrieval. Retrieval is conventionally measured by Recall@$k$, which is blind to a required passage climbing the ranking without crossing the top-$k$ boundary and thus too sparse a training signal. We instead relax the indicator into a soft membership. Writing $\mathrm{rank}(d;\mathcal{G})$ for the position of passage $d$ under retrieval on $\mathcal{G}$, the score $s(q;\mathcal{G})$ of Eq. (1) is instantiated as

$$s(q;\mathcal{G}) \;=\; \frac{1}{|\mathcal{P}^\star|}\sum_{d\in\mathcal{P}^\star} u\big(\mathrm{rank}(d;\mathcal{G})\big), \qquad u(r) \;=\; \begin{cases} 1, & r < k, \\ \big(\frac{k}{r+1+\delta}\big)^{\gamma}, & r \ge k, \end{cases} \tag{6}$$

**Proposition 2.1.** *For every $\gamma > 0$ and $\delta \ge 0$, the score in Eq. (6) satisfies*

$$\mathrm{Recall@}k \;\le\; s(q;\mathcal{G}) \;\le\; \mathrm{Recall@}k \;+\; u(k)\big(1-\mathrm{Recall@}k\big),$$

*and $s(q;\mathcal{G}) = 1$ if and only if $\mathcal{P}^\star \subseteq \mathcal{R}_k(q;\mathcal{G})$. In particular, $s(q;\mathcal{G}) \to \mathrm{Recall@}k$ as $\gamma \to \infty$.*

Due to the space limit, the proof of Proposition 2.1 is deferred to Appendix A.1.

**Evolver Reward.** The evolver is optimized by a composite reward with three feedback terms, a *retrieval gain* reward $R_{\mathsf{gain}}$, a *generalization* reward $R_{\mathsf{gen}}$ and an *operation cost* $R_{\mathsf{ops}}$, all gated by a *format* check $\mathbb{1}_{\mathsf{fmt}} \in \{0,1\}$. For a well-formed round, the reward is

$$R\big(O_t^{(r)}\big) \;=\; R_{\mathsf{gain}} \;+\; w_{\mathsf{gen}}\, R_{\mathsf{gen}} \;-\; w_{\mathsf{ops}}\, R_{\mathsf{ops}}, \tag{7}$$

where $w_{\mathsf{gen}}$ and $w_{\mathsf{ops}}$ balance the two regularizing terms.

*Format reward.* A round that violates the output schema (Appendix E), or edits passages outside its view, is malformed: it bypasses graph evaluation and receives $R\big(O_t^{(r)}\big) = -\lambda$ in place of Eq. (7).

*Retrieval gain reward.* For operations $\mathcal{A}$ applied on top of earlier rounds, write $\Delta(q;\mathcal{A}) = s\big(q;\mathcal{G}_t \oplus O_t^{<r} \oplus \mathcal{A}\big) - s\big(q;\mathcal{G}_t \oplus O_t^{<r}\big)$ for the change it brings to the score of a query $q$. The gain reward $R_{\mathsf{gain}} = \Delta\big(q_t; O_t^{(r)}\big)$ then measures the round's direct benefit on the source query, taken as an increment so that a round is credited for what it adds rather than what earlier rounds already secured.

*Generalization reward.* Since operations are consolidated into the shared graph, an edit should generalize beyond the query that triggered it. We take $R_{\mathsf{gen}}$ to be the aggregate score change over the neighboring queries $\mathcal{N}(q_t)$, those whose retrieved passages overlap with $q_t$'s. This term internalizes side effects on the shared structure into the reward, so that edits overfitting the source query at the expense of its neighbors are directly penalized.

*Operation cost.* The rewards above score the union of a round's operations, so an operation that accomplishes nothing is carried by the ones that do. Writing $\mathcal{N}^+(q_t) = \{q_t\} \cup \mathcal{N}(q_t)$ for the queries under evaluation, we apply each operation on its own and count those that move none of them, $R_{\mathsf{ops}} = \big|\{\, o \in O_t^{(r)} : \Delta(q;\{o\}) = 0 \text{ for all } q \in \mathcal{N}^+(q_t)\,\}\big|$. The count is waived when the union achieves more than any operation alone, so that cooperation is not mistaken for waste.

**Extractor Reward.** The extractor is optimized by a composite reward with a *coverage* reward $R_{\mathsf{cov}}$, a *selection cost* $R_{\mathsf{cost}}$ and an *evolution* reward $R_{\mathsf{evo}}$, gated by the same format check $\mathbb{1}_{\mathsf{fmt}}$. For a well-formed trajectory, the reward is

$$R\big(\mathcal{S}_t\big) \;=\; R_{\mathsf{cov}} \;-\; w_{\mathsf{cost}}\, R_{\mathsf{cost}} \;+\; w_{\mathsf{evo}}\, R_{\mathsf{evo}}, \tag{8}$$

*Coverage reward.* The extractor moves no passage; what it decides is which passages the evolver may touch. We therefore credit it with how much of the room left on this query its selection covers. Writing $s^+(q_t;\mathcal{S})$ for the score $q_t$ would attain if every passage in $\mathcal{S}$ were brought inside the top-$k$,

$$R_{\mathsf{cov}} \;=\; \frac{s^+(q_t;\mathcal{S}_t) \;-\; s(q_t;\mathcal{G}_t)}{1 \;-\; s(q_t;\mathcal{G}_t)}, \tag{9}$$

the fraction of that room which $\mathcal{S}_t$ places within the evolver's reach, set to 1 when the query is solved.

*Selection cost.* The term above is maximized by keeping every passage, so we charge a constant amount for each one retained beyond a free allowance $m$, $R_{\mathsf{cost}} = \max\big(0,\, |\mathcal{S}_t| - m\big)$, which keeps the extractor from retaining passages that add no coverage.

*Evolution gain reward.* Whether extraction was good ultimately depends on whether the evolver could act on it, so we return the realized gain to the extractor, $R_{\mathsf{evo}} = s\big(q_t;\mathcal{G}_t \oplus O_t\big) - s\big(q_t;\mathcal{G}_t\big)$. However, the reward cannot be used on its own: early in training, the evolver has limited capability, so the gain reflects its own limits and the two dense terms must provide the effective training signal.

**Credit Assignment.** The two modules differ in whether an intermediate decision can be scored on its own. A round of evolution is applied and followed by a fresh retrieval, so it forms an independent sample, $\hat{A}_t^{(r)} = R\big(q_t, O_t^{(r)}\big) - b^{(r)}$ against the baseline $b^{(r)}$ of the rollouts for $q_t$. It is credited with the gain it realized and not with later increments, whose noise would otherwise decide the sign of the first and most consequential edit. Since later rounds inherit less room, we remove the round effect before centering. The extractor admits no such measurement: it moves no passage, and selecting a required passage is not yet evidence that the selection helped. Its trajectory takes one terminal reward, shared by every turn, $\hat{A}_t = R(\mathcal{S}_t) - \frac{1}{G}\sum_i R(\mathcal{S}_t^i)$, averaged over trajectories so that length does not weigh on the baseline. Credit crosses the modules once: the terminal reward returns the evolver's realized gain to the extractor, while the evolver is judged only on the subgraph it received.

**Optimization.** Both policies are trained with GRPO (Shao et al., 2024). For the $G$ rollouts $\{y_i\}_{i=1}^{G}$ sampled on a query with advantages $\hat{A}_i$ defined above, each module maximizes

$$\mathcal{J}(\theta) \;=\; \mathbb{E}\Big[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{j=1}^{|y_i|}\min\big(\rho_{i,j}\hat{A}_i,\; \mathrm{clip}(\rho_{i,j},\,1-\epsilon,\,1+\epsilon)\,\hat{A}_i\big)\Big] \;-\; \beta\,\mathbb{D}_{\mathrm{KL}}\big[\pi_\theta \,\|\, \pi_{\mathsf{ref}}\big], \tag{10}$$

The two policies are LoRA adapters over a shared frozen backbone, and they are updated in alternating phases rather than together. Only one adapter receives gradient in a phase; the other keeps producing its half of the trajectory with frozen weights, so each module is optimized against a fixed partner while still seeing the inputs that partner actually generates.

# 3 THEORETICAL ANALYSIS

To understand whether evolving the graph along the query stream approaches the optimal graph of Eq. (1), we analyze the evolution process theoretically.

**Problem setup.** Consider a finite memory corpus $D$ and the initial graph $\mathcal{G}_0$ built from it. Let $\mathbb{G}$ denote the graphs reachable from $\mathcal{G}_0$ via the operations of Sec. 2.2. We assume queries $q_0, q_1, \ldots$ arrive as i.i.d. draws from a distribution $\mathcal{Q}$ with finite support $\mathcal{S} = \{q^{(1)}, \ldots, q^{(n)}\}$, where each $q \in \mathcal{S}$ has frequency $\pi_q > 0$; this models deployment, where a memory is queried repeatedly. With the score $s(q; \mathcal{G})$ of Eq. (6), we measure the quality of a graph by

$$F(\mathcal{G}) := \mathbb{E}_{q\sim\mathcal{Q}}\big[s(q;\mathcal{G})\big] = \sum_{q\in\mathcal{S}} \pi_q\, s(q;\mathcal{G}), \qquad F^\star := \sup_{\mathcal{G}\in\mathbb{G}} F(\mathcal{G}), \tag{11}$$

and, for each query, define its ceiling $\bar{s}(q) := \sup_{\mathcal{G}\in\mathbb{G}} s(q;\mathcal{G})$ and its room $\delta(q;\mathcal{G}) := \bar{s}(q) - s(q;\mathcal{G}) \in [0,1]$ under $\mathcal{G}$. We write $g_t(q) := s(q;\mathcal{G}_{t+1}) - s(q;\mathcal{G}_t)$ for the change in the score of $q$. At step $t$ the model proposes $\mathcal{O}_t$ on $(q_t, \mathcal{G}_t)$ and $\mathcal{G}_{t+1} = \mathcal{G}_t \oplus \mathcal{O}_t^+$ with $\mathcal{O}_t^+$ given by Eq. (5), where we take $\mathcal{N}(q_t) = \mathcal{S}$ so that no query regresses; the neighbor set of Sec. 2.2 is its computational approximation.

**Definition 3.1** ($\kappa$-progress)**.** Let $q \in \mathcal{S}$ be a query and $\mathcal{G} \in \mathbb{G}$ a graph where $q$ has room to improve, i.e. $\delta(q;\mathcal{G}) > 0$. The model makes *$\kappa$-progress* if there exists $\kappa \in (0,1]$ such that, for every such pair, the verified set $\mathcal{O}^+$ produced on $(q,\mathcal{G})$ closes at least a $\kappa$ fraction of the room in expectation,

$$\mathbb{E}\big[s(q;\mathcal{G}\oplus\mathcal{O}^+) - s(q;\mathcal{G}) \,\big|\, q,\mathcal{G}\big] \;\geq\; \kappa\,\delta(q;\mathcal{G}). \tag{12}$$

**Theorem 3.2** (Convergence)**.** *Under Eq. (5), if the model makes $\kappa$-progress, then $F(\mathcal{G}_t)$ is nondecreasing in the step $t$, and for every $T \geq 0$ the expected gap to the optimum after $T$ steps satisfies*

$$\mathbb{E}\big[F^\star - F(\mathcal{G}_T)\big] \;\leq\; \sum_{q\in\mathcal{S}} \pi_q\big(1-\kappa\,\pi_q\big)^T \delta(q;\mathcal{G}_0) \;\leq\; \sum_{q\in\mathcal{S}} \pi_q\big(1-\kappa\,\pi_q\big)^T. \tag{13}$$

*Remark* 3.3. Theorem 3.2 shrinks each query's initial room by a factor $(1-\kappa\pi_q)$ per step. For a uniform stream over $n$ queries the gap is at most $e^{-\kappa T/n}$. Policy quality enters only through $\kappa$, the fraction of remaining room closed per verified update. The proof is in Appendix A.2. Corollary A.3 shows that when repairs transfer within a conversation, $n$ is replaced by the number of conversations, so $T/n$ becomes the number of queries per conversation.

# 4 EXPERIMENT

## 4.1 EXPERIMENTAL SETUP

**Datasets and Metrics.** We evaluate on *four* long-term memory benchmarks spanning *six* evaluation settings: LoCoMo (Maharana et al., 2024), LongMemEval-S (Wu et al., 2025a), PersonaMem (Jiang et al., 2025) at 32K and 128K context scales, and PrefEval (Zhao et al., 2025) under explicit and implicit preference queries. The first two are free-form QA, scored by F1 and LLM-Judge macro-averaged over question categories so that no single category dominates; the other two are four-way multiple choice, scored by accuracy and MRR, the reciprocal rank of the gold option under the model's likelihood over option letters. Details are in Appendix B.1.

**Baselines.** We compare against 16 memory or graph baselines in four groups. *Two context methods*: LongContext, which packs raw history up to the window limit, and Naive RAG, which retrieves top-$K$ dialogue chunks from a dense index. *Four graph-based RAG methods*: GraphRAG (Edge et al., 2024), LightRAG (Guo et al., 2024), HippoRAG2 (Gutiérrez et al., 2025), and HyperGraphRAG (Luo et al., 2026a). *Seven training-free memory methods*: A-Mem (Xu et al., 2025b), Mem0 (Chhikara et al., 2025), MemGAS (Xu et al., 2025a), MemoryOS (Kang et al., 2025), LightMem (Fang et al., 2025), SimpleMem (Liu et al., 2026a), and MRAgent (Ji et al., 2026). *Three training-based memory methods*: MemAgent (Yu et al., 2025), AutoGraph-R1 (Tsang et al., 2026), and MemCoE (Xu et al., 2026), which learn retrieval or memory-update policies with RL. Details are in Appendix B.2.

**Implementation Details.** All methods use the same Qwen3-8B backbone for memory construction and answer generation, so that differences reflect the memory design rather than the LLM. Retrieval operates at the chunk level, where a chunk typically consists of three consecutive turns and a turn is one user–assistant exchange. Chunks are represented with Contriever embeddings, and the top-3 chunks are passed to the QA model, with gpt-5-mini judging free-form answers. The training-based baselines are also initialized from Qwen3-8B and trained on the same 500 queries, evenly sampled at random from LoCoMo and PersonaMem, using 4 NVIDIA H20 GPUs; LongMemEval-S and PrefEval are never trained on and thus stay out-of-distribution. Details are in Appendix B.3.

Table 1: Main results on four long-term memory benchmarks with 16 baselines. Higher is better for all metrics. The best and second-best results in each column are shaded.

| Method | LoCoMo | | LongMemEval-S | | PersonaMem | | | | PrefEval | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 32K | | 128K | | Explicit | | Implicit | |
| | F1 | Judge | F1 | Judge | ACC | MRR | ACC | MRR | ACC | MRR | ACC | MRR |
| *Context Methods* | | | | | | | | | | | | |
| LongContext | 28.06 | 44.98 | 9.86 | 11.17 | 51.78 | 70.67 | 39.05 | 62.08 | 48.10 | 66.22 | 42.80 | 63.40 |
| Naive RAG | 26.14 | 39.06 | 30.47 | 40.44 | 54.33 | 71.96 | 43.16 | 64.73 | 69.80 | 80.92 | 54.60 | 70.40 |
| *Graph-based RAG* | | | | | | | | | | | | |
| GraphRAG | 14.38 | 21.73 | 24.43 | 35.13 | 49.75 | 69.31 | 40.81 | 63.30 | 37.30 | 59.41 | 36.70 | 59.25 |
| LightRAG | 19.40 | 29.05 | 30.66 | 41.99 | 53.48 | 71.60 | 43.16 | 64.91 | 61.10 | 75.17 | 48.70 | 67.44 |
| HippoRAG2 | 29.78 | 45.31 | 32.90 | 43.11 | 54.67 | 72.68 | 44.96 | 65.95 | 69.70 | 81.01 | 54.60 | 70.39 |
| HyperGraphRAG | 28.92 | 41.39 | 34.33 | 46.54 | 56.03 | 73.42 | 44.92 | 65.99 | 58.90 | 73.59 | 49.60 | 67.89 |
| *Training-free Memory* | | | | | | | | | | | | |
| A-Mem | 26.14 | 39.87 | 30.91 | 39.71 | 54.67 | 72.30 | 43.56 | 65.06 | 69.40 | 80.27 | 54.00 | 69.98 |
| Mem0 | 22.22 | 37.59 | 28.76 | 38.70 | 53.48 | 71.52 | 45.10 | 66.03 | 60.20 | 74.52 | 56.40 | 71.35 |
| MemGAS | 26.39 | 38.38 | 33.16 | 44.15 | 53.14 | 71.36 | 44.70 | 65.51 | 68.80 | 80.33 | 55.50 | 71.16 |
| MemoryOS | 29.25 | 43.87 | 34.36 | 45.15 | 56.71 | 72.33 | 46.42 | 66.83 | 65.80 | 78.84 | 54.00 | 70.93 |
| LightMem | 19.27 | 27.78 | 28.05 | 35.33 | 51.61 | 71.29 | 41.80 | 64.43 | 64.70 | 79.03 | 47.40 | 66.86 |
| SimpleMem | 31.27 | 46.80 | 34.12 | 44.86 | 51.27 | 70.77 | 42.90 | 64.74 | 70.20 | 82.22 | 50.20 | 68.73 |
| MRAgent | 31.48 | 46.79 | 22.21 | 27.45 | 51.10 | 70.91 | 40.89 | 64.01 | 64.20 | 77.60 | 51.10 | 69.27 |
| *Training-based Memory* | | | | | | | | | | | | |
| MemAgent | 28.63 | 49.96 | 32.64 | 40.51 | 56.03 | 70.57 | 43.75 | 63.16 | 57.80 | 71.77 | 48.80 | 66.01 |
| AutoGraph-R1 | 30.20 | 46.08 | 35.72 | 45.90 | 51.78 | 70.80 | 43.97 | 65.37 | 56.80 | 72.21 | 56.30 | 71.27 |
| MemCoE | 31.41 | 45.37 | 33.50 | 44.21 | 56.54 | 72.99 | 46.83 | 67.20 | 61.50 | 73.50 | 51.70 | 67.39 |
| **G-Evolver** | 37.48 | 58.02 | 36.44 | 48.57 | 58.74 | 75.89 | 49.17 | 68.87 | 78.00 | 86.08 | 58.70 | 72.81 |

### 4.2 Main Results

To examine whether evolving the graph improves both retrieval and downstream QA, we compare **G-Evolver** with 16 baselines on four long-term memory benchmarks in Tables 1 and 2.

**QA performance.** Table 1 presents the QA results. Naive RAG is a competitive baseline, outperforming GraphRAG and LightRAG on most columns, and only HippoRAG2 and HyperGraphRAG consistently surpass it outside PrefEval, suggesting that a graph helps only when its structure is sound. Several training-free memory systems are competitive on PrefEval but none exceeds 47 on LoCoMo Judge, which requires multi-hop and temporal reasoning across sessions. Training-based methods do not resolve this: MemAgent ranks second on LoCoMo Judge and MemCoE on PersonaMem-128K, but both fall behind even Naive RAG on every PrefEval metric, and AutoGraph-R1, which learns the triple extractor with RL, performs only on par with the untrained HippoRAG2, confirming that learning within-passage extraction does not reach cross-passage structure. **G-Evolver** is best across all benchmarks and metrics, with the largest margins on LoCoMo and PrefEval-Explicit: it improves LoCoMo Judge by 8.06 over the strongest baseline, while other gains are smaller but consistent.

**Retrieval performance.** Table 2 presents the retrieval results, where **G-Evolver**'s gains are mostly larger than in QA, reaching 15.66 R@3 on LoCoMo over the strongest baseline. Since **G-Evolver** shares its retriever and initial graph with HippoRAG2, the 20.85 gain over it stems from the provisional edits. Comparing the two tables also clarifies where the remaining errors lie. On PrefEval-Explicit, retrieval is nearly saturated and QA gains most. On PrefEval-Implicit, a gain of over ten points in R@3 yields only 2.30 in ACC, indicating that inferring implicit preferences is limited by the reader rather than by memory. On PersonaMem-128K, retrieval remains the main bottleneck for all methods, and the QA gains are correspondingly bounded.

### 4.3 Generalization with Different Initial Graphs

**G-Evolver** is instantiated on HippoRAG2 by default, so we examine whether its gains depend on that particular graph and retrieval algorithm by applying it to GraphRAG and LightRAG, which differ in both construction and propagation, reusing the policies trained on HippoRAG2 without retraining. Table 3 shows consistent improvements on every benchmark and metric for all three graphs. The gains grow as the initial graph weakens: GraphRAG, the weakest baseline, improves the most, with LoCoMo Judge rising by over 30 points, and after evolution both GraphRAG and LightRAG match or exceed the static HippoRAG2 on most columns. Evolution thus compensates for defects in the initial construction rather than relying on a strong starting graph, and the learned policies transfer across graphs, so **G-Evolver** can serve as a plug-in refinement layer for existing graph memories.

Table 2: Retrieval results on four memory benchmarks with 13 baselines. R@$k$ denotes Recall@$k$. The best and second-best per column are shaded. Methods without a retrieval stage are omitted.

| Method | LoCoMo | | LongMemEval-S | | PersonaMem | | | | PrefEval | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 32K | | 128K | | Explicit | | Implicit | |
| | R@3 | R@5 | R@3 | R@5 | R@3 | R@5 | R@3 | R@5 | R@3 | R@5 | R@3 | R@5 |
| *Retrieval-augmented Generation* | | | | | | | | | | | | |
| Naive RAG | 58.26 | 68.03 | 77.00 | 82.20 | 33.62 | 43.12 | 11.81 | 16.17 | 76.50 | 82.30 | 74.30 | 82.50 |
| GraphRAG | 20.90 | 31.97 | 73.40 | 81.60 | 23.26 | 32.43 | 9.39 | 14.56 | 54.20 | 63.80 | 52.70 | 63.10 |
| LightRAG | 33.99 | 49.40 | 80.80 | 86.40 | 31.07 | 46.69 | 11.95 | 17.53 | 81.60 | 83.80 | 60.60 | 73.50 |
| HippoRAG2 | 66.01 | 74.52 | 80.40 | 85.20 | 38.88 | 48.73 | 12.58 | 17.79 | 92.30 | 96.50 | 74.50 | 82.50 |
| HyperGraphRAG | 59.47 | 69.59 | 86.40 | 90.40 | 35.65 | 46.86 | 12.69 | 18.63 | 61.10 | 71.60 | 71.30 | 80.00 |
| *Memory Systems* | | | | | | | | | | | | |
| A-Mem | 58.21 | 67.52 | 77.20 | 82.60 | 34.30 | 42.61 | 11.88 | 16.46 | 91.70 | 95.80 | 74.40 | 81.80 |
| Mem0 | 66.77 | 75.63 | 84.60 | 88.40 | 39.05 | 53.31 | 14.96 | 20.87 | 80.40 | 86.50 | 85.10 | 92.10 |
| MemGAS | 52.57 | 65.86 | 83.20 | 86.00 | 31.24 | 44.82 | 13.57 | 18.70 | 93.50 | 97.40 | 73.40 | 81.20 |
| MemoryOS | 64.85 | 73.92 | 78.40 | 83.40 | 37.69 | 49.58 | 12.61 | 18.85 | 67.90 | 68.60 | 72.80 | 84.50 |
| LightMem | 49.70 | 58.36 | 85.40 | 89.00 | 36.50 | 46.52 | 11.55 | 16.36 | 63.70 | 69.00 | 62.70 | 70.60 |
| SimpleMem | 70.09 | 79.76 | 83.20 | 86.60 | 37.86 | 48.90 | 14.41 | 19.77 | 71.80 | 77.10 | 61.80 | 72.40 |
| MRAgent | 71.20 | 81.92 | 77.00 | 83.80 | 33.45 | 44.48 | 11.95 | 16.72 | 88.90 | 93.40 | 68.20 | 79.70 |
| AutoGraph-R1 | 69.39 | 79.36 | 84.20 | 87.40 | 28.86 | 44.14 | 11.37 | 16.65 | 71.50 | 78.60 | 78.50 | 86.00 |
| **G-Evolver** | 86.86 | 91.84 | 91.00 | 92.80 | 42.95 | 60.44 | 16.83 | 22.63 | 97.50 | 99.70 | 95.70 | 98.10 |

Table 3: Generalization of **G-Evolver** with different initial graphs. Each block reports a static-graph baseline, the same graph after evolution by **G-Evolver**, and the absolute improvement.

| Method | LoCoMo | | LongMemEval-S | | PersonaMem | | | | PrefEval | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 32K | | 128K | | Explicit | | Implicit | |
| | F1 | Judge | F1 | Judge | ACC | MRR | ACC | MRR | ACC | MRR | ACC | MRR |
| GraphRAG | 14.38 | 21.73 | 24.43 | 35.13 | 49.75 | 69.31 | 40.81 | 63.30 | 37.30 | 59.41 | 36.70 | 59.25 |
| +**G-Evolver** | 36.46 | 54.61 | 35.40 | 46.48 | 56.88 | 73.84 | 47.60 | 67.79 | 69.70 | 80.92 | 54.40 | 71.41 |
| *Improv.* | +22.08 | +32.88 | +10.97 | +11.35 | +7.13 | +4.53 | +6.79 | +4.49 | +32.40 | +21.51 | +17.70 | +12.16 |
| LightRAG | 19.40 | 29.05 | 30.66 | 41.99 | 53.48 | 71.60 | 43.16 | 64.91 | 61.10 | 75.17 | 48.70 | 67.44 |
| +**G-Evolver** | 34.30 | 50.06 | 34.31 | 44.81 | 55.86 | 73.46 | 46.53 | 67.26 | 68.10 | 79.93 | 55.40 | 71.93 |
| *Improv.* | +14.90 | +21.01 | +3.65 | +2.82 | +2.38 | +1.86 | +3.37 | +2.35 | +7.00 | +4.76 | +6.70 | +4.49 |
| HippoRAG2 | 29.78 | 45.31 | 32.90 | 43.11 | 54.67 | 72.68 | 44.96 | 65.95 | 69.70 | 81.01 | 54.60 | 70.39 |
| +**G-Evolver** | 37.48 | 58.02 | 36.44 | 48.57 | 58.74 | 75.89 | 49.17 | 68.87 | 78.00 | 86.08 | 58.70 | 72.81 |
| *Improv.* | +7.70 | +12.71 | +3.54 | +5.46 | +4.07 | +3.21 | +4.21 | +2.92 | +8.30 | +5.07 | +4.10 | +2.42 |

### 4.4 Ablation Study

**Rewards and credit assignment.** To verify the contribution of each reward term and of hierarchical credit assignment in GEPO, we ablate them individually on LoCoMo and LongMemEval-S (Table 4). The extractor is trained with the coverage reward (CR) and the evolution gain (EG), and the evolver with the retrieval gain (RG) and the generalization reward (GR). Each term is removed in turn, and hierarchical credit assignment is further replaced with a single outcome reward shared by both modules. Every variant underperforms the full model on all metrics. The retrieval gain and the coverage reward, the dense terms that directly score each module's decisions, matter most, as removing them costs 8.5 and 6.6 F1 on LoCoMo, respectively. The generalization reward and the evolution gain bring smaller but consistent gains, confirming the value of both regularizing terms. Removing credit assignment drops F1 by 4.6 and 5.0, showing that crediting each module for the outcome it is responsible for is essential.

Table 4: Ablation of rewards and credit assignment in GEPO. CR and EG belong to the extractor; RG and GR belong to the evolver.

| Variant | LoCoMo | | LongMemEval-S | |
|---|---|---|---|---|
| | F1 | Judge | F1 | Judge |
| **G-Evolver** | **37.48** | **58.02** | **36.44** | **48.57** |
| *Extractor Reward* | | | | |
| w/o CR | 30.84 | 53.21 | 29.21 | 43.21 |
| w/o EG | 33.90 | 55.52 | 33.18 | 45.80 |
| *Evolver Reward* | | | | |
| w/o RG | 28.99 | 51.53 | 27.27 | 41.43 |
| w/o GR | 34.01 | 55.21 | 32.81 | 46.03 |
| *Credit Assignment* | | | | |
| w/o Credit | 32.85 | 54.93 | 31.46 | 45.10 |

**Backbone LLMs.** To examine whether the gains of **G-Evolver** depend on a particular backbone, we instantiate HippoRAG2 with three LLMs, Qwen3-8B, GPT-4o-mini, and Gemini-3-Flash, each used to construct the initial graph and generate answers, and apply the same trained policies without retraining. As shown in Figure 3, **G-Evolver** improves the Judge score of every backbone on both benchmarks. No backbone alone matches even the weakest evolved variant, whereas graph evolution consistently lifts all of them, indicating that the improvement stems from the repaired structure rather than any specific backbone.

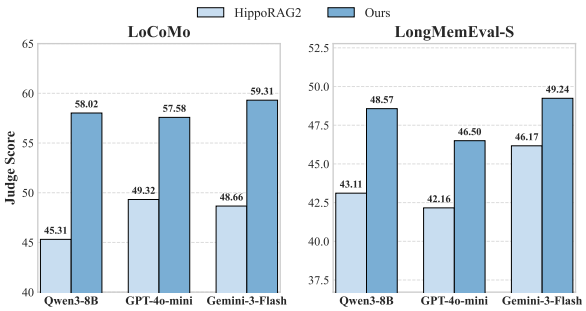



Figure 3: Judge scores of HippoRAG2 and **G-Evolver** with different backbone LLMs.

### 4.5 Efficiency Analysis

To assess the efficiency of **G-Evolver**, we measure the average inference time per query of all methods on LoCoMo under the same backbone and plot it against the Judge score in Figure 4. **G-Evolver** answers a query in under two seconds, adding less than one second over HippoRAG2, and remains faster than agentic memory systems such as MemAgent, MRAgent, and SimpleMem, which are also at least 8 points less accurate. This overhead remains acceptable because both modules stop adaptively, with the extractor skipping evolution when retrieval already suffices and the evolver abstaining once the view is sound. **G-Evolver** thus offers the best trade-off between accuracy and latency.

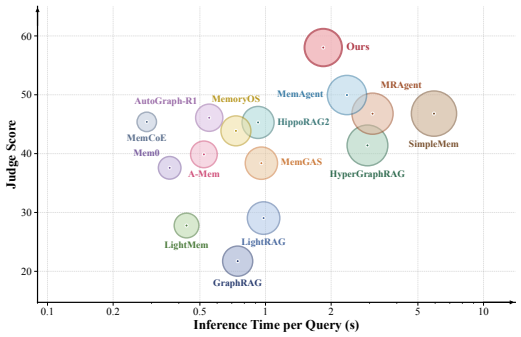



Figure 4: Judge score versus inference time per query on LoCoMo. Bubble size denotes time variance across queries.

## 5 Related Work

**Graph-based Memory for LLM Agents.** A growing line of work organizes agent memory into graphs and performs retrieval by traversing them (Wang et al., 2024; Shu et al.; Zhang et al., 2026a). One class builds entity-centric knowledge graphs: Zep (Rasmussen et al., 2025) attaches temporal validity to edges. Another class links memory units directly: A-Mem (Xu et al., 2025b) stores notes and establishes links among related notes upon insertion. Despite differing granularity, both classes fix the graph structure at write time with heuristic linking rules: when retrieval fails, no mechanism corrects the responsible structure, and defects silently persist. **G-Evolver** instead treats retrieval as a source of structural feedback, repairing the graph continuously as queries arrive.

**Reinforcement Learning for Memory Agents.** Another line applies reinforcement learning to memory systems (Wang et al., 2025; Zhu et al., 2026; Liu et al., 2026b). One class optimizes graph construction or retrieval: Graph-R1 (Luo et al., 2026b) learns an agentic retrieval policy over the graph. Another class learns memory management: Memory-R1 (Yan et al., 2025) trains an LLM to decide entry-level operations such as adding, updating, or deleting memories. In both classes, however, the reward shapes model parameters rather than the serving-time structure, so structural defects exposed by individual queries remain uncorrected. In contrast, GEPO directs verification feedback to the structure itself, learning to extract and evolve policies that edit the graph online.

## 6 Conclusion

We present **G-Evolver**, which turns graph memory from a frozen topology into a structure that evolves continuously with the query stream. An extractor localizes the query-relevant subgraph, an evolver edits it provisionally before retrieval, and edits verified by retrieval feedback are consolidated into the persistent graph. We train both modules with GEPO, which converts the verification signal into hierarchical credit over extraction and evolution decisions. We further prove that, under a repairability condition, the expected retrieval score converges geometrically to its optimum. Experiments on long-term memory benchmarks show that **G-Evolver** consistently outperforms static-graph baselines. In future work, we will add pruning and merging operations and handle non-stationary query streams.

### AI USE STATEMENT

In this work, generative AI tools were used in two ways. First, we used LLMs to assist in checking the theoretical analysis in Section 3 and Appendix A, including verifying the derivation steps and the consistency of notation; all theorems and proofs were formulated and written by the authors, and every AI-flagged issue was independently verified before any revision. Second, we used generative AI tools to edit the manuscript for readability, including grammar correction, sentence-level rephrasing, and condensing text to fit the page limit; all suggested edits were reviewed by the authors. These constitute the only uses of generative AI in this work. We take responsibility for the final content, including text and derivations checked or edited with the aid of generative AI.

### REPRODUCIBILITY STATEMENT

We have made the following efforts to ensure reproducibility. Our code, including the training and inference pipelines of **`G-Evolver`** and the scripts for all experiments, is anonymously released at https://anonymous.4open.science/r/G-Evolver-C15E/README.md. The method is specified in Section 2, with the reward definitions, credit assignment, and optimization objective given in Section 2.3, and the prompts of the extractor and the evolver provided in Appendix E. All assumptions underlying the theoretical results are stated in Section 3, and complete proofs are given in Appendix A. The benchmarks, metrics, baselines, and implementation details, including the backbone, retriever, chunking, and training configuration, are described in Section 4.1 and Appendix B.

## REFERENCES

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

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. Lightmem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*, 2025.

Jizhan Fang, Buqiang Xu, Zhixian Wang, Haoliang Cao, Xinle Deng, Baohua Dong, Hangcheng Zhu, Ruohui Huang, Gang Yu, Ying Wei, et al. Rethinking memory as continuously evolving connectivity. *arXiv preprint arXiv:2605.28773*, 2026.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: neurobiologically inspired long-term memory for large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pp. 59532–59569, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*, 2025.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, et al. Evermemos: A self-organizing memory operating system for structured long-horizon reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 45836–45853, 2026.

Zhengjun Huang, Zhoujin Tian, Qintian Guo, Fangyuan Zhang, Yingli Zhou, Di Jiang, Zeying Xie, and Xiaofang Zhou. Licomemory: Lightweight and cognitive agentic memory for efficient long-term reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 36842–36858, 2026.

Shuo Ji, Yibo Li, and Bryan Hooi. Memory is reconstructed, not retrieved: Graph memory for llm agents. *arXiv preprint arXiv:2606.06036*, 2026.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J Taylor, and Dan Roth. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. *arXiv preprint arXiv:2504.14225*, 2025.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. Magma: A multi-graph based agentic memory architecture for ai agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36848–36865, 2026.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25972–25981, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, et al. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*, 2025.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026a.

Jiaqi Liu, Xinyu Ye, Peng Xia, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Evolvemem: Self-evolving memory architecture via autoresearch for llm agents. *arXiv preprint arXiv:2605.13941*, 2026b.

Haoran Luo, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, et al. Hypergraphrag: Retrieval-augmented generation via hypergraph-structured knowledge representation. *Advances in Neural Information Processing Systems*, 38:152206–152234, 2026a.

Haoran Luo, Haihong E, Guanting Chen, Qika Lin, Yikai Guo, Fangzhi Xu, Zemin Kuang, Meina Song, Xiaobao Wu, Yifan Zhu, and Luu Anh Tuan. Graph-r1: Towards agentic graphrag framework via end-to-end reinforcement learning, 2026b. URL https://arxiv.org/abs/2507.21892.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. Nemori: Self-organizing agent memory inspired by cognitive science. *arXiv preprint arXiv:2508.03341*, 2025.

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

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xKDZAW0He3.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yiheng Shu, Saisri Padmaja Jonnalagedda, Xiang Gao, Bernal Jiménez Gutiérrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. Remem: Reasoning with episodic memory in language agent. In *The Fourteenth International Conference on Learning Representations*.

Hong Ting Tsang, Jiaxin Bai, Haoyu Huang, Qiao Xiao, Tianshi Zheng, Baixuan Xu, Shujie Liu, and Yangqiu Song. Autograph-r1: End-to-end reinforcement learning for knowledge graph construction. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23351–23374, 2026.

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

Zheng Wang, Zhongyang Li, Zeren Jiang, Dandan Tu, and Wei Shi. Crafting personalized agents through retrieval-augmented generation on editable memory graphs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4891–4906, 2024.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=pZiyCaVuti.

Yaxiong Wu, Yongyue Zhang, Sheng Liang, and Yong Liu. Sgmem: Sentence graph memory for long-term conversational agents. *arXiv preprint arXiv:2509.21212*, 2025b.

Zhaofen Wu, Hanrong Zhang, Fulin Lin, Wujiang Xu, Xinran Xu, Yankai Chen, Henry Peng Zou, Shaowen Chen, Weizhi Zhang, Xue Liu, et al. Gam: Hierarchical graph-based agentic memory for llm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34647–34664, 2026.

Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, Tong Xu, et al. From single to multi-granularity: Toward long-term memory association and selection of conversational agents. *arXiv preprint arXiv:2505.19549*, 2025a.

Derong Xu, Shuochen Liu, Pengfei Luo, Pengyue Jia, Yingyi Zhang, Yi Wen, Yimin Deng, Wenlin Zhang, Enhong Chen, Xiangyu Zhao, et al. Learning how and what to memorize: Cognition-inspired two-stage optimization for evolving memory. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 44987–45011, 2026.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025b.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

Chongrui Ye, Yuxiang Liu, Yu Wang, Haofei Yu, Yining Zhao, Ge Liu, Julian McAuley, and Jiaxuan You. Auto-dreamer: Learning offline memory consolidation for language agents. *arXiv preprint arXiv:2605.20616*, 2026.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025.

Jiawei Yu, Yixiang Fang, Xilin Liu, and Yuchi Ma. H-mem: A novel memory mechanism for evolving and retrieving agent memory via a hybrid structure. *arXiv preprint arXiv:2605.15701*, 2026a.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, 2026b.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *Advances in Neural Information Processing Systems*, 38:12988–13018, 2026a.

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. *arXiv preprint arXiv:2602.02474*, 2026b.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026c.

Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. Memory as action: Autonomous context curation for long-horizon agentic tasks. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 19149–19164, 2026d.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. Do llms recognize your preferences? evaluating personalized preference following in llms. In *13th International Conference on Learning Representations, ICLR 2025*, pp. 72531–72574. International Conference on Learning Representations, ICLR, 2025.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024.

Chulun Zhou, Chunkang Zhang, Guoxin Yu, Fandong Meng, Jie Zhou, Wai Lam, and Mo Yu. Hgmem: Hypergraph-based working memory to improve multi-step rag for long-context complex relational modeling. *arXiv preprint arXiv:2512.23959*, 2025a.

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

Baoheng Zhu, Deyu Bo, Delvin Ce Zhang, and Xiao Wang. Graph-grpo: Training graph flow models with reinforcement learning. *arXiv preprint arXiv:2603.10395*, 2026.

# Technical Appendix

## Table of Contents

**A Theoretical Proofs and Extensions** **15**
A.1 Proof of Proposition 2.1 . . . 15
A.2 Proof of Theorem 3.2 . . . 16
A.3 Convergence under Transfer . . . 16

**B Dataset and Baseline Details** **17**
B.1 Dataset and Metric Details . . . 17
B.2 Baselines Details . . . 18
B.3 Implementation Details . . . 20

**C Extended Experiments** **20**
C.1 Category-level Performance . . . 20
C.2 Validation Performance during Training . . . 21
C.3 Graph Evolution with the Query Stream . . . 21

**D Extended Related Work** **22**
D.1 Memory for LLM Agents . . . 22
D.2 Graph-based Memory for LLM Agents . . . 23

**E Prompts** **23**

# A Theoretical Proofs and Extensions

## A.1 Proof of Proposition 2.1

*Proposition 2.1.* For every $\gamma > 0$ and $\delta \geq 0$, the score in Eq. (6) satisfies

$$\text{Recall@}k \;\leq\; s(q;\mathcal{G}) \;\leq\; \text{Recall@}k \;+\; u(k)\big(1 - \text{Recall@}k\big),$$

and $s(q;\mathcal{G}) = 1$ if and only if $\mathcal{P}^\star \subseteq \mathcal{R}_k(q;\mathcal{G})$. In particular, $s(q;\mathcal{G}) \to \text{Recall@}k$ as $\gamma \to \infty$.

*Proof.* Fix a query $q$ and a graph $\mathcal{G}$. Write $\rho := \text{Recall@}k(q;\mathcal{G}) = |\mathcal{P}^\star \cap \mathcal{R}_k(q;\mathcal{G})|/|\mathcal{P}^\star|$, abbreviate $r_d := \text{rank}(d;\mathcal{G})$, and set

$$t \;:=\; \frac{k}{k+1+\delta} \;\in\; (0,1). \tag{14}$$

Ranks are 0-based, so that $\mathcal{R}_k(q;\mathcal{G}) = \{d : r_d < k\}$; a required passage the retriever never surfaces is assigned the truncation depth, which is at least $k$, so every $r_d$ is a finite integer and $u(r_d)$ is well defined. Recall that $k \geq 1$, $\mathcal{P}^\star \neq \emptyset$, $\gamma > 0$ and $\delta \geq 0$.

For $r \geq k$ we have $0 < k < k+1+\delta \leq r+1+\delta$, hence $0 < \frac{k}{r+1+\delta} \leq t < 1$. Since $x \mapsto x^\gamma$ is strictly increasing on $(0,\infty)$,

$$0 \;<\; u(r) \;\leq\; u(k) \;=\; t^\gamma \;<\; 1 \qquad \text{for every } r \geq k. \tag{15}$$

Moreover $r \mapsto \frac{k}{r+1+\delta}$ is strictly decreasing, so $u$ is strictly decreasing on $\{r \geq k\}$. Together with $u \equiv 1$ on $\{r < k\}$, this shows that $u$ is non-increasing on $\{0,1,2,\dots\}$ and that

$$u(r) = 1 \iff r < k. \tag{16}$$

Split $\mathcal{P}^\star = \mathcal{H} \sqcup \mathcal{M}$ with $\mathcal{H} := \mathcal{P}^\star \cap \mathcal{R}_k(q;\mathcal{G})$, so that $|\mathcal{H}| = \rho\,|\mathcal{P}^\star|$ and $|\mathcal{M}| = (1-\rho)\,|\mathcal{P}^\star|$. By Eq. (16), $u(r_d) = 1$ for $d \in \mathcal{H}$, whence

$$s(q;\mathcal{G}) \;=\; \rho \;+\; \frac{1}{|\mathcal{P}^\star|} \sum_{d \in \mathcal{M}} u(r_d). \tag{17}$$

Every $d \in \mathcal{M}$ has $r_d \geq k$, so by Eq. (15) each summand lies in $(0, t^\gamma]$, and Eq. (17) gives

$$\rho \;\leq\; s(q;\mathcal{G}) \;\leq\; \rho + t^\gamma(1-\rho), \tag{18}$$

which is the claimed sandwich since $u(k) = t^\gamma$.

If $\rho < 1$, then by Eq. (18) and $t^\gamma < 1$, $s(q;\mathcal{G}) \leq \rho + t^\gamma(1-\rho) < \rho + (1-\rho) = 1$. If $\rho = 1$, then $\mathcal{M} = \emptyset$ and Eq. (17) gives $s(q;\mathcal{G}) = 1$. Hence $s(q;\mathcal{G}) = 1$ if and only if $\rho = 1$, i.e. if and only if $\mathcal{P}^\star \subseteq \mathcal{R}_k(q;\mathcal{G})$.

From Eq. (18) and $1 - \rho \leq 1$,

$$0 \;\leq\; s(q;\mathcal{G}) - \rho \;\leq\; t^\gamma. \tag{19}$$

Because $t \in (0,1)$, $t^\gamma \to 0$ as $\gamma \to \infty$, so $s(q;\mathcal{G}) \to \rho = \text{Recall@}k$. The bound in Eq. (19) depends only on $k$, $\delta$ and $\gamma$, not on $q$ or $\mathcal{G}$, so the convergence is uniform over queries and graphs, at a rate exponential in $\gamma$. □

**Corollary A.1** (Sensitivity below the top-$k$)**.** *Let $\mathcal{G}$ and $\mathcal{G}'$ satisfy $r'_d \leq r_d$ for every $d \in \mathcal{P}^\star$, and suppose that no required passage crosses the top-$k$ boundary, i.e. there is no $d \in \mathcal{P}^\star$ with $r'_d < k \leq r_d$. If $r'_{d_0} < r_{d_0}$ for some $d_0 \in \mathcal{P}^\star$ with $r'_{d_0} \geq k$, then*

$$\text{Recall@}k(q;\mathcal{G}') = \text{Recall@}k(q;\mathcal{G}), \qquad \textit{while} \qquad s(q;\mathcal{G}') > s(q;\mathcal{G}).$$

*Proof.* Since $r'_d \leq r_d$, the implication $r_d < k \Rightarrow r'_d < k$ is automatic, and the boundary condition supplies the converse; hence $\mathcal{P}^\star \cap \mathcal{R}_k(q;\mathcal{G}') = \mathcal{P}^\star \cap \mathcal{R}_k(q;\mathcal{G})$ and the two recalls coincide. As shown in the proof of Proposition 2.1, $u$ is non-increasing, so $u(r'_d) \geq u(r_d)$ for every $d \in \mathcal{P}^\star$; and $u$ is strictly decreasing on $\{r \geq k\}$, which contains both $r'_{d_0}$ and $r_{d_0} > r'_{d_0} \geq k$, so $u(r'_{d_0}) > u(r_{d_0})$. Averaging over $\mathcal{P}^\star$ gives the strict inequality. □

### A.2 Proof of Theorem 3.2

*Theorem* 3.2 (Convergence). Under Eq. (5), if the model makes $\kappa$-progress, then $F(\mathcal{G}_t)$ is nondecreasing in the step $t$, and for every $T \geq 0$ the expected gap to the optimum after $T$ steps satisfies

$$\mathbb{E}\big[F^\star - F(\mathcal{G}_T)\big] \;\leq\; \sum_{q\in\mathcal{S}} \pi_q \big(1-\kappa\,\pi_q\big)^T \delta(q;\mathcal{G}_0) \;\leq\; \sum_{q\in\mathcal{S}} \pi_q \big(1-\kappa\,\pi_q\big)^T.$$

*Proof.* With $\mathcal{N}(q_t) = \mathcal{S}$, every set accepted by Eq. (5) is checked as a whole against $\mathcal{G}_t$ on all of $\mathcal{S}$, so the consolidated set satisfies $s(q;\mathcal{G}_t \oplus \mathcal{O}_t^+) \geq s(q;\mathcal{G}_t)$ for every $q \in \mathcal{S}$, and trivially so when $\mathcal{O}_t^+ = \emptyset$. Hence

$$g_t(q) \;\geq\; 0 \qquad \text{for all } t \text{ and all } q \in \mathcal{S}, \tag{20}$$

which already gives that $F(\mathcal{G}_t) = \sum_q \pi_q s(q;\mathcal{G}_t)$ is nondecreasing in $t$. The bound is proved in three steps.

*Step 1: the gap is bounded by the room.* For every $\mathcal{G}' \in \mathbb{G}$ and $q \in \mathcal{S}$, $s(q;\mathcal{G}') \leq \bar{s}(q)$ by the definition of the ceiling, so $F(\mathcal{G}') \leq \sum_q \pi_q \bar{s}(q)$; taking the supremum over $\mathcal{G}'$ gives $F^\star \leq \sum_q \pi_q \bar{s}(q)$, and subtracting $F(\mathcal{G}) = \sum_q \pi_q s(q;\mathcal{G})$ yields

$$F^\star - F(\mathcal{G}) \;\leq\; \sum_{q\in\mathcal{S}} \pi_q\, \delta(q;\mathcal{G}) \qquad \text{for every } \mathcal{G} \in \mathbb{G}. \tag{21}$$

*Step 2: expected progress on the triggering query.* Let $H_t := (\mathcal{G}_0, q_0, \mathcal{O}_0^+, \ldots, q_{t-1}, \mathcal{O}_{t-1}^+)$ be the history before step $t$. Then $\mathcal{G}_t = \mathcal{G}_0 \oplus \mathcal{O}_0^+ \oplus \cdots \oplus \mathcal{O}_{t-1}^+$ is a function of $H_t$; $q_t$ is independent of $H_t$ because the stream is i.i.d.; and, given $(q_t, \mathcal{G}_t)$, the set $\mathcal{O}_t^+$ is generated by the model without further dependence on $H_t$, so $g_t(q_t) = s(q_t;\mathcal{G}_t \oplus \mathcal{O}_t^+) - s(q_t;\mathcal{G}_t)$ has, conditionally on $(H_t, q_t)$, the distribution in Definition 3.1. We claim that

$$\mathbb{E}\big[g_t(q_t) \,\big|\, H_t, q_t\big] \;\geq\; \kappa\, \delta(q_t;\mathcal{G}_t). \tag{22}$$

If $\delta(q_t;\mathcal{G}_t) = 0$, then $s(q_t;\mathcal{G}_t) = \bar{s}(q_t) \geq s(q_t;\mathcal{G}_{t+1})$, which with Eq. (20) forces $g_t(q_t) = 0$, so both sides of Eq. (22) vanish. If $\delta(q_t;\mathcal{G}_t) > 0$, Eq. (22) is exactly Eq. (12) applied to $(q_t, \mathcal{G}_t)$.

*Step 3: per-query recurrence.* Fix $q \in \mathcal{S}$. By the definition of $g_t$, $\delta(q;\mathcal{G}_{t+1}) = \delta(q;\mathcal{G}_t) - g_t(q)$. On the event $\{q_t = q\}$, Eq. (22) gives $\mathbb{E}[g_t(q) \mid H_t, q_t] \geq \kappa\, \delta(q;\mathcal{G}_t)$; on $\{q_t \neq q\}$, Eq. (20) gives $\mathbb{E}[g_t(q) \mid H_t, q_t] \geq 0$. Hence

$$\mathbb{E}\big[\delta(q;\mathcal{G}_{t+1}) \,\big|\, H_t, q_t\big] \;\leq\; \delta(q;\mathcal{G}_t) - \kappa\, \mathbb{1}\{q_t = q\}\, \delta(q;\mathcal{G}_t).$$

Since $q_t$ is independent of $H_t$ with $\mathbb{P}(q_t = q) = \pi_q$, while $\delta(q;\mathcal{G}_t)$ is a function of $H_t$, averaging over $q_t$ gives

$$\mathbb{E}\big[\delta(q;\mathcal{G}_{t+1}) \,\big|\, H_t\big] \;\leq\; \big(1-\kappa\,\pi_q\big)\, \delta(q;\mathcal{G}_t). \tag{23}$$

Taking expectations on both sides, $\mathbb{E}[\delta(q;\mathcal{G}_{t+1})] \leq (1-\kappa\pi_q)\, \mathbb{E}[\delta(q;\mathcal{G}_t)]$, and iterating from $t = 0$, where $\delta(q;\mathcal{G}_0)$ is deterministic, yields $\mathbb{E}[\delta(q;\mathcal{G}_T)] \leq (1-\kappa\pi_q)^T \delta(q;\mathcal{G}_0)$. Apply Eq. (21) with $\mathcal{G} = \mathcal{G}_T$, take expectations, and sum over the finitely many $q$. The last inequality in Eq. (13) uses $\delta(q;\mathcal{G}_0) \leq 1$. □

### A.3 Convergence under Transfer

Consolidated operations live in the shared graph, so a repair made for one query may serve other queries over the same memory. In this subsection we ask how the convergence rate changes if a repair made for the triggering query carries over, at some proportion, to the queries of the same conversation.

**Assumption A.2** ($\beta$-transferability)**.** Let $\mathcal{C}(q) \subseteq \mathcal{S}$ denote the queries over the same conversation as $q$. Since each conversation forms an independent memory, these sets partition $\mathcal{S}$ and repairs made for $q$ affect no query outside $\mathcal{C}(q)$. The model is *$\beta$-transferable* if there exists $\beta \in (0, 1]$ such that a repair made for a query benefits its conversation, on average, by at least $\beta$ of its own gain, i.e., for every $q \in \mathcal{S}$ and $\mathcal{G} \in \mathbb{G}$,

$$\mathbb{E}\Big[\mathbb{E}_{q'\sim\mathcal{C}(q)}\big[s(q';\mathcal{G}\oplus\mathcal{O}^+) - s(q';\mathcal{G})\big]\Big] \;\geq\; \beta\, \mathbb{E}\big[s(q;\mathcal{G}\oplus\mathcal{O}^+) - s(q;\mathcal{G})\big], \tag{24}$$

where $\mathcal{O}^+$ is the verified operation set returned by the model on $(q, \mathcal{G})$. Under Eq. (5) no query regresses, so Eq. (24) always holds with $\beta = 1/|\mathcal{C}(q)|$; the assumption is informative when $\beta$ is well above this floor.

**Corollary A.3** (Transfer Convergence)**.** *Suppose the model makes $\kappa$-progress and is $\beta$-transferable. Let $\mathcal{C}_1, \dots, \mathcal{C}_m$ be the distinct conversations, with frequencies $\pi_j := \mathcal{Q}(\mathcal{C}_j)$ and average rooms $D_t(j) := \mathbb{E}_{q\sim\mathcal{C}_j}[\delta(q; \mathcal{G}_t)]$. Then for every $T \geq 0$,*

$$\mathbb{E}\big[F^\star - F(\mathcal{G}_T)\big] \;\leq\; \sum_{j=1}^{m} \pi_j \big(1 - \beta\kappa\,\pi_j\big)^T D_0(j) \;\leq\; \sum_{j=1}^{m} \pi_j \big(1 - \beta\kappa\,\pi_j\big)^T. \tag{25}$$

*In particular, for $m$ equally frequent conversations the gap is at most $e^{-\beta\kappa T/m}$, so the factor $n$ of Theorem 3.2 is replaced by the number of conversations, and $T/m$ is the number of queries each conversation has received.*

*Proof.* Let $H_t$ be the history before step $t$, as in the proof of Theorem 3.2. By the definition of $g_t$,

$$D_{t+1}(j) \;=\; D_t(j) - \mathbb{E}_{q\sim\mathcal{C}_j}\big[g_t(q)\big]. \tag{26}$$

On the event $\{q_t \in \mathcal{C}_j\}$, we have $\mathcal{C}(q_t) = \mathcal{C}_j$, so Eq. (24) applied to $(q_t, \mathcal{G}_t)$ and then Eq. (22) give

$$\mathbb{E}\big[D_{t+1}(j) \,\big|\, H_t, q_t\big] \;\leq\; D_t(j) - \beta\,\mathbb{E}\big[g_t(q_t) \,\big|\, H_t, q_t\big] \;\leq\; D_t(j) - \beta\kappa\,\delta(q_t; \mathcal{G}_t), \tag{27}$$

while on $\{q_t \notin \mathcal{C}_j\}$, Eq. (20) and Eq. (26) give

$$\mathbb{E}\big[D_{t+1}(j) \,\big|\, H_t, q_t\big] \;\leq\; D_t(j). \tag{28}$$

Combining Eq. (27) and Eq. (28),

$$\mathbb{E}\big[D_{t+1}(j) \,\big|\, H_t, q_t\big] \;\leq\; D_t(j) - \beta\kappa\mathbb{1}\{q_t \in \mathcal{C}_j\}\,\delta(q_t; \mathcal{G}_t). \tag{29}$$

Since $q_t$ is independent of $H_t$ and $\delta(\cdot\,; \mathcal{G}_t)$ is a function of $H_t$,

$$\mathbb{E}\big[\mathbb{1}\{q_t \in \mathcal{C}_j\}\,\delta(q_t; \mathcal{G}_t) \,\big|\, H_t\big] \;=\; \sum_{q\in\mathcal{C}_j} \pi_q\,\delta(q; \mathcal{G}_t) \;=\; \pi_j\,D_t(j), \tag{30}$$

and averaging over $q_t$ yields

$$\mathbb{E}\big[D_{t+1}(j) \,\big|\, H_t\big] \;\leq\; \big(1 - \beta\kappa\,\pi_j\big)\,D_t(j). \tag{31}$$

Taking expectations and iterating from $t = 0$,

$$\mathbb{E}\big[D_T(j)\big] \;\leq\; \big(1 - \beta\kappa\,\pi_j\big)^T D_0(j). \tag{32}$$

Finally, since the conversations partition $\mathcal{S}$, Eq. (21) gives

$$F^\star - F(\mathcal{G}_T) \;\leq\; \sum_{q\in\mathcal{S}} \pi_q\,\delta(q; \mathcal{G}_T) \;=\; \sum_{j=1}^{m} \pi_j\,D_T(j); \tag{33}$$

taking expectations and inserting Eq. (32) proves Eq. (25). The last inequality there uses $D_0(j) \leq 1$, and the uniform case follows from $(1-x)^T \leq e^{-xT}$ with $\pi_j = 1/m$. □

# B Dataset and Baseline Details

## B.1 Dataset and Metric Details

Table 5 summarizes the statistics of the four benchmarks.

**LoCoMo.** LoCoMo (Maharana et al., 2024) is a long-conversation memory benchmark consisting of 10 richly annotated multi-session dialogues. Each conversation spans roughly 27 sessions and 300 turns, yielding a haystack of approximately 22K tokens. The benchmark provides 1,986 questions categorized into five types: single-hop factual recall, temporal reasoning, multi-hop inference, open-domain questions that require synthesizing information across sessions, and adversarial queries designed to test robustness against unanswerable or misleading questions. This diversity makes LoCoMo a comprehensive testbed for evaluating fine-grained conversational memory over extended interactions.

**LongMemEval.** LongMemEval (Wu et al., 2025a) evaluates long-term memory across six core abilities: single-session user information extraction, single-session assistant information extraction, multi-session reasoning, temporal reasoning, knowledge update tracking, and preference recall. It contains 500 questions, each paired with a synthetic conversation history assembled from real-world chat logs. We use the S variant, whose histories average 92 sessions and 124K tokens.

**PersonaMem.** PersonaMem (Jiang et al., 2025) focuses on personalized memory that tracks evolving user preferences across multi-session dialogues. Each conversation is grounded in a detailed user persona and spans multiple sessions in which the user's preferences gradually shift. The benchmark defines seven question types: recalling user-shared facts, providing preference-aligned recommendations, suggesting new ideas, recalling reasons behind preference updates, tracking full preference evolution, generalizing to new scenarios, and recalling facts mentioned by the user. We evaluate on two context scales: PersonaMem-32K (37 conversations, 589 questions, ∼37K average tokens) and PersonaMem-128K (60 conversations, 2,727 questions, ∼162K average tokens), enabling analysis of how context length affects preference-aware reasoning.

**PrefEval.** PrefEval (Zhao et al., 2025) is designed to measure whether a model can identify and respect user preferences embedded in long conversation histories. Each sample consists of approximately 51 distractor sessions interleaved with one or two gold sessions that contain the target preference, resulting in an average context of roughly 55K tokens. The benchmark offers two complementary splits: PrefEval-Explicit (1,000 queries), where preferences are stated directly by the user, and PrefEval-Implicit (1,000 queries), where preferences must be inferred from the user's behavior or indirect statements. Together, these splits test both surface-level and deep preference comprehension under realistic, noisy retrieval conditions.

**Metrics.** Let $\mathcal{Q}$ denote the set of test questions. For free-form answers (LoCoMo and LongMemEval-S), questions are grouped into categories $\mathcal{C}$ (Table 6), and both metrics are macro-averaged over categories so that no single category dominates. Writing $\mathcal{Q}_c$ for the questions of category $c$, a per-question score $m_q$ is aggregated as

$$M = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{Q}_c|} \sum_{q \in \mathcal{Q}_c} m_q. \tag{34}$$

*F1* measures token overlap between the predicted answer $\hat{a}_q$ and the gold answer $a_q$ after normalization,

$$m_q^{\mathrm{F1}} = \frac{2\,P_q R_q}{P_q + R_q}, \qquad P_q = \frac{|\hat{a}_q \cap a_q|}{|\hat{a}_q|}, \quad R_q = \frac{|\hat{a}_q \cap a_q|}{|a_q|}, \tag{35}$$

where $|\hat{a}_q \cap a_q|$ counts overlapping tokens. *Judge* scores a question as correct if an LLM judge deems the predicted answer correct given the gold answer,

$$m_q^{\mathrm{Judge}} = \mathbb{1}\big[\mathrm{LLM}(q, \hat{a}_q, a_q) = \texttt{correct}\big]. \tag{36}$$

For multiple-choice questions (PersonaMem and PrefEval), let $\mathrm{rank}_q$ denote the position of the correct option in the model's ranking of the candidate options. We report

$$\mathrm{ACC} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbb{1}\big[\mathrm{rank}_q = 1\big], \qquad \mathrm{MRR} = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{\mathrm{rank}_q}. \tag{37}$$

Retrieval quality is measured against the gold passages $\mathcal{P}_q^\star$ annotated for each question. Writing $\mathcal{R}_k(q)$ for the top-$k$ retrieved passages, we report

$$\mathrm{Recall@}k = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{|\mathcal{P}_q^\star \cap \mathcal{R}_k(q)|}{|\mathcal{P}_q^\star|}, \tag{38}$$

### B.2 Baselines Details

We describe the 16 baseline methods used in our experiments, grouped into the four categories of Table 1: context methods, graph-based RAG, training-free memory, and training-based memory.

Table 5: Statistics of the four benchmarks. "#Sess." is the average number of sessions per conversation and "Avg. Tokens" the average history length.

| Benchmark | #Conv. | #Sess. | Avg. Tokens | #Questions |
|---|---|---|---|---|
| LoCoMo (Maharana et al., 2024) | 10 | 27 | 22K | 1,986 |
| LongMemEval-S (Wu et al., 2025a) | 500 | 92 | 124K | 500 |
| PersonaMem-32K (Jiang et al., 2025) | 37 | 32 | 37K | 589 |
| PersonaMem-128K (Jiang et al., 2025) | 60 | 142 | 162K | 2,727 |
| PrefEval-Explicit (Zhao et al., 2025) | 1,000 | 51 | 54K | 1,000 |
| PrefEval-Implicit (Zhao et al., 2025) | 1,000 | 52 | 56K | 1,000 |

- **LongContext** directly feeds as much raw interaction history as fits into the model context window, relying solely on the backbone's native long-context capacity without any external indexing.
- **Naive RAG** (Lewis et al., 2020) indexes all historical dialogue snippets in a vector store and retrieves the top-$K$ segments most relevant to the query via dense similarity, which are then concatenated into the prompt for generation.
- **GraphRAG** (Edge et al., 2024) uses an LLM to build a two-stage graph index: it first derives an entity knowledge graph from the documents, then generates community summaries for related groups of entities. At query time, each community summary produces a partial response, which are aggregated into a final answer.
- **LightRAG** (Guo et al., 2024) incorporates graph structures into text indexing and employs a dual-level retrieval paradigm that combines low-level entity-centric retrieval with high-level thematic retrieval, and supports incremental index updates for efficient adaptation to new data.
- **HippoRAG2** (Gutiérrez et al., 2025) extracts entity–relation triples, links passages through shared entities, and retrieves via Personalized PageRank over a graph that includes both entity and passage nodes, enabling single-step multi-hop retrieval across passages.
- **HyperGraphRAG** (Luo et al., 2026a) extends graph-based RAG beyond binary relations by representing $n$-ary relational facts through hyperedges, with a hypergraph construction method, a hypergraph retrieval strategy, and hypergraph-guided generation.
- **A-Mem** (Xu et al., 2025b) organizes memories following the Zettelkasten method. For each new memory, it generates a structured note with descriptions, keywords, and tags, links it to related historical memories, and can further trigger updates to existing ones.
- **Mem0** (Chhikara et al., 2025) extracts salient facts from each conversation turn and consolidates them into a memory store through add, update, delete, and no-op operations decided by an LLM.
- **MemGAS** (Xu et al., 2025a) builds multi-granularity memory units, associates new memories with historical ones via Gaussian mixture clustering, and selects the retrieval granularity per query with an entropy-based router.
- **MemoryOS** (Kang et al., 2025) draws on operating-system memory management principles to build a hierarchical storage architecture with short-term, mid-term, and long-term personal memory, coordinated by storage, updating, retrieval, and generation modules with segmented page organization and FIFO-based updates.
- **LightMem** (Fang et al., 2025) is inspired by the Atkinson–Shiffrin model and organizes memory into three stages: a sensory memory that pre-compresses and topically groups raw input, a topic-aware short-term memory that consolidates these groups, and a long-term memory with a sleep-time update mechanism that decouples consolidation from online inference.
- **SimpleMem** (Liu et al., 2026a) compresses interactions into multi-view memory units, synthesizes related context online to remove redundancy, and infers query intent to plan retrieval scope.
- **MRAgent** (Ji et al., 2026) represents memory as a cue–tag–content graph and replaces one-shot retrieval with active reconstruction, iteratively exploring and pruning retrieval paths based on evidence accumulated during reasoning.
- **MemAgent** (Yu et al., 2025) reads text segment by segment and updates a fixed-length token memory through an overwrite strategy, trained end-to-end with a multi-conversation extension of DAPO to retain answer-critical information while discarding distractors.

- **AutoGraph-R1** (Tsang et al., 2026) trains the LLM graph constructor with end-to-end RL, rewarding a graph by its downstream utility in a RAG pipeline, either as a knowledge carrier or as a retrieval index, rather than by extraction accuracy.
- **MemCoE** (Xu et al., 2026) is a two-stage framework that first learns a memory organization guideline from contrastive feedback and then trains a guideline-aligned RL policy that decides what to store and update in the memory bank.

### B.3 Implementation Details

Each conversation forms an independent memory, which is split into chunks that serve as retrieval units. For LoCoMo, whose conversations are short, each chunk is a whole session; for LongMemEval-S, PersonaMem and PrefEval, each chunk consists of three consecutive turns, where a turn is one user–assistant exchange. All dense retrieval components encode chunks and queries with Contriever, and every method returns its top-3 chunks; methods that store finer-grained units are mapped back to chunks until three distinct chunks are covered. Qwen3-8B is the backbone for memory construction and answer generation in all methods, except that the end-to-end RL baselines (MemAgent and MemCoE) answer with their own trained models. All methods share the same prompts on each dataset. LoCoMo and LongMemEval-S follow the evaluation setting of SimpleMem (Liu et al., 2026a), and adversarial (LoCoMo) and abstention (LongMemEval-S) questions are excluded from the averages; PersonaMem and PrefEval are evaluated as multiple choice. Free-form answers are judged by `gpt-5-mini-2025-08-07`. The Gemini-3-Flash model in our ablation is the preview release (`gemini-3-flash-preview`), since no stable version has been released.

All training-based baselines are initialized from Qwen3-8B and trained with standard GRPO on the same 500 training queries for 150 steps, with 8 queries per step and 4 rollouts per query. Each method uses the reward from its original paper and otherwise keeps its official configuration, e.g., a learning rate of $10^{-6}$ and a KL coefficient of $10^{-3}$ for MemAgent and MemCoE.

The evolver starts from the HippoRAG2 index of each conversation, built with its default settings. In all main experiments, operations are applied provisionally within each query and are not consolidated across queries, so no test label is used at inference; the effect of consolidation along the query stream is analyzed separately in Appendix C.3. The extractor selects evidence passages from the top-25 retrieved passages. The evolver then edits the graph with three operations: adding a triple, modifying an existing triple, and adding an edge between two passages, over at most $R = 3$ rounds (Eq. (4)) to keep its cost bounded. For the retrieval score (Eq. (6)), we set $k = 3$, $\gamma = 1$ and $\delta = 4$; with $\gamma = 1$, any $\delta > k - 1$ ensures that entering the top-$k$ outweighs any rank gain below it. Every rollout, during training and validation alike, starts from the initial graph of its conversation, and its edits are discarded afterward; each validation query in Appendix C.2 is thus evaluated on its own provisional repair, never on edits made for other queries. Both modules share the Qwen3-8B backbone, each with its own LoRA adapter (rank 64), and they are trained alternately. Each 50-step cycle updates the evolver for 40 steps and then the extractor for 10 steps while the other adapter stays frozen; three cycles give 150 steps in total. We use GRPO with 4 rollouts per query, a learning rate of $10^{-5}$ and a KL coefficient of 0.01. For the evolver reward (Eq. (7)), we set $w_{\text{gen}} = 0.1$, $w_{\text{ops}} = 0.05$ and $\lambda = 1$, and scale $R_{\text{gain}} + w_{\text{gen}} R_{\text{gen}}$ by 10 so that a successful edit clearly outweighs the format penalty. For the extractor reward (Eq. (8)), we set $w_{\text{cost}} = 0.05$, $m = 2$ and $w_{\text{evo}} = 1$.

## C Extended Experiments

### C.1 Category-level Performance

To analyze where the gains of **G-Evolver** come from, we break down the Judge score by question category on LoCoMo and LongMemEval-S for all methods, as shown in Table 6. The largest improvements appear exactly on the categories that require linking scattered evidence. On LoCoMo, **G-Evolver** improves temporal reasoning by 9.4 points and multi-hop by 2.1 points over the strongest baseline, and on LongMemEval-S it leads multi-session by 5.0 points, which is consistent with our motivation that shared-entity linking misses the causal and temporal relations across passages. Single-hop questions also benefit substantially, by 11.1 points, and single-session-user questions to a smaller extent, since inserting missing triples makes previously unreachable passages retrievable. The remaining gaps point to limitations outside the graph. MemAgent leads open-domain questions by reading the entire history rather than retrieving, and knowledge-update questions favor methods that surface the latest fact, whereas the current operations of **G-Evolver** never remove outdated structure, which motivates the pruning operation left for future work. Finally, single-session-preference

Table 6: Per-category Judge scores (%) on LoCoMo and LongMemEval-S. LoCoMo categories: multi-hop (MH), temporal reasoning (TR), open-domain knowledge (OD), and single-hop (SH). LongMemEval-S categories: temporal reasoning (TR), knowledge update (KU), multi-session (MS), single-session user (SS-U), single-session assistant (SS-A), and single-session preference (SS-P). Overall is the macro-average over categories. The best result per column is shaded

| Method | LoCoMo | | | | | LongMemEval-S | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MH | TR | OD | SH | Overall | TR | KU | MS | SS-U | SS-A | SS-P | Overall |
| *Context Methods* | | | | | | | | | | | | |
| LongContext | 43.62 | 34.58 | 36.46 | 65.28 | 44.98 | 11.02 | 27.78 | 4.13 | 6.25 | 17.86 | 0.00 | 11.17 |
| Naive RAG | 38.30 | 31.78 | 27.08 | 59.10 | 39.06 | 20.47 | 63.89 | 22.31 | 48.44 | 87.50 | 0.00 | 40.44 |
| *Graph-based RAG* | | | | | | | | | | | | |
| GraphRAG | 26.24 | 11.21 | 21.88 | 27.59 | 21.73 | 17.32 | 62.50 | 25.62 | 50.00 | 55.36 | 0.00 | 35.13 |
| LightRAG | 31.56 | 15.58 | 27.08 | 41.97 | 29.05 | 14.96 | 52.78 | 23.97 | 64.06 | 92.86 | 3.33 | 41.99 |
| HippoRAG2 | 42.55 | 39.25 | 33.33 | 66.11 | 45.31 | 16.54 | 62.50 | 23.14 | 65.62 | 87.50 | 3.33 | 43.11 |
| HyperGraphRAG | 41.13 | 29.28 | 29.17 | 65.99 | 41.39 | 18.90 | 63.89 | 27.27 | 76.56 | 89.29 | 3.33 | 46.54 |
| *Training-free Memory* | | | | | | | | | | | | |
| A-Mem | 37.59 | 32.71 | 30.21 | 58.98 | 39.87 | 20.47 | 56.94 | 23.14 | 48.44 | 89.29 | 0.00 | 39.71 |
| Mem0 | 37.94 | 21.50 | 32.29 | 58.62 | 37.59 | 14.17 | 59.72 | 24.79 | 78.12 | 55.36 | 0.00 | 38.70 |
| MemGAS | 37.94 | 32.71 | 27.08 | 55.77 | 38.38 | 21.26 | 58.33 | 27.27 | 65.62 | 85.71 | 6.67 | 44.15 |
| MemoryOS | 45.04 | 33.96 | 37.50 | 58.98 | 43.87 | 21.26 | 55.56 | 31.40 | 70.31 | 85.71 | 6.67 | 45.15 |
| LightMem | 23.05 | 27.10 | 27.08 | 33.89 | 27.78 | 19.68 | 62.50 | 28.93 | 76.56 | 14.29 | 10.00 | 35.33 |
| SimpleMem | 44.68 | 41.74 | 28.12 | 72.65 | 46.80 | 19.68 | 58.33 | 35.54 | 73.44 | 82.14 | 0.00 | 44.86 |
| MRAgent | 44.68 | 39.88 | 31.25 | 71.34 | 46.79 | 15.75 | 55.56 | 14.88 | 57.81 | 10.71 | 10.00 | 27.45 |
| *Training-based Memory* | | | | | | | | | | | | |
| MemAgent | 54.96 | 26.79 | 43.75 | 74.32 | 49.96 | 21.26 | 58.33 | 26.45 | 64.06 | 69.64 | 3.33 | 40.51 |
| AutoGraph-R1 | 43.97 | 33.33 | 35.42 | 71.58 | 46.08 | 21.26 | 58.33 | 35.54 | 78.12 | 82.14 | 0.00 | 45.90 |
| MemCoE | 49.29 | 20.87 | 38.54 | 72.77 | 45.37 | 21.26 | 52.78 | 33.88 | 68.75 | 78.57 | 10.00 | 44.21 |
| **G-Evolver** | 57.09 | 51.09 | 38.54 | 85.37 | 58.02 | 22.83 | 61.11 | 40.50 | 81.25 | 85.71 | 0.00 | 48.57 |

questions are near zero for every method, indicating that inferring implicit preferences is bounded by the reader rather than by memory, in line with our observation on PrefEval-Implicit.

### C.2 Validation Performance during Training

To examine whether GEPO learns efficiently and stably, we track validation retrieval performance during training and compare it with AutoGraph-R1, which also trains a graph constructor with RL from the same backbone. Every 25 steps we evaluate Recall@3 on 128 queries sampled from PersonaMem-32K, which are disjoint from the 500 training queries, as shown in Figure 5. **G-Evolver** starts from about 35% because it evolves the existing HippoRAG2 graph. **G-Evolver** improves by about 9 points within the first 50 steps and plateaus around 45% after step 75, while AutoGraph-R1 requires 100 steps to reach its peak of 32% and remains more than 13 points behind throughout training. Both curves fluctuate mildly after step 100, and **G-Evolver** keeps the gap stable, indicating that evolving an existing graph provides both a stronger starting point and faster, more stable learning than constructing one with RL.

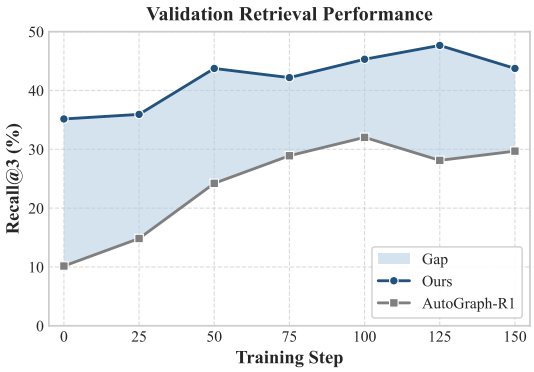



Figure 5: Validation Retrieval Performance of **G-Evolver** and AutoGraph-R1 during RL training, evaluated every 25 steps on 128 queries sampled from PersonaMem-32K.

### C.3 Graph Evolution with the Query Stream

To verify the advantage of graph evolution over LLM reranking, we compare the two on top of the same HippoRAG2 graph, on LoCoMo and PersonaMem where multiple queries share a conversation. LLM rerank re-scores the top-25 retrieved passages of each query, the same candidates seen by the

Table 7: Graph evolution vs. LLM reranking on HippoRAG2. Avg. Δ: mean gain over HippoRAG2; †: uses gold labels.

| Method | LoCoMo | | PersonaMem | | | | Avg. Δ |
|---|---|---|---|---|---|---|---|
| | | | 32K | | 128K | | |
| | F1 | Judge | ACC | MRR | ACC | MRR | |
| HippoRAG2 | 29.78 | 45.31 | 54.67 | 72.68 | 44.96 | 65.95 | – |
| *Per-query refinement* | | | | | | | |
| + Rerank w/ GPT-5-mini | 31.99 | 48.65 | 55.35 | 73.22 | 45.43 | 66.28 | +1.26 |
| + Rerank w/ GPT-5.2 | 36.93 | 57.17 | 57.56 | 74.96 | 48.29 | 68.26 | +4.97 |
| + **G-Evolver** | 37.48 | 58.02 | 58.74 | 75.89 | 49.17 | 68.87 | +5.80 |
| *Continual refinement* | | | | | | | |
| + **G-Evolver** w/ GPT-5-mini judge | 38.43 | 59.49 | 59.59 | 76.56 | 49.91 | 69.37 | +6.67 |
| + **G-Evolver** w/ gold labels† | 39.89 | 62.32 | 62.31 | 78.71 | 52.00 | 70.82 | +8.78 |

extractor, with GPT-5-mini or GPT-5.2. **G-Evolver** is evaluated in two settings: *per-query*, the setting of all main experiments, where the graph is repaired for the current query and the edits are then discarded, and *continual*, where repairs that pass the consolidation rule in Eq.( 5) are committed to the graph before later queries are served, following the two-batch protocol of Appendix C.3. Table 7 reports the results. Under the per-query setting, reranking with GPT-5-mini brings only a modest gain (+1.26 on average), and only the stronger GPT-5.2 (+4.97) reaches a level comparable to **G-Evolver** (+5.80), which achieves this with a single repair and remains ahead on PersonaMem. The limitation of reranking, however, is that its improvement is bound to the query at hand: a ranking corrects the output of one query but leaves the graph and its defects unchanged, so nothing is passed on to the queries that follow. Graph evolution has no such ceiling. Consolidating repairs verified by GPT-5-mini, the same model that gains little as a reranker, improves **G-Evolver** on every metric, e.g., by 1.47 Judge on LoCoMo, and the average gain rises further to 8.78 with gold labels. Verified repairs thus accumulate in the graph, and later queries no longer re-encounter the defects fixed by earlier ones, as intended by the continual evolution design.

# D Extended Related Work

## D.1 Memory for LLM Agents

Long-term memory lets LLM agents retain information beyond the context window. Early systems keep interaction records in an external store and retrieve them by similarity, adding periodic reflection (Park et al., 2023), forgetting-curve updating (Zhong et al., 2024), or OS-style paging between context and storage (Packer et al., 2023). Later work imposes richer organization. MemOS (Li et al., 2025) and EverMemOS (Hu et al., 2026) manage memory as operating-system resources with hierarchical, self-organizing units, Nemori (Nan et al., 2025) segments experience into episodes following cognitive principles, and H-Mem (Yu et al., 2026a) couples a temporal–semantic tree with an entity knowledge graph. MAGMA (Jiang et al., 2026) represents each memory item in separate semantic, temporal, causal, and entity graphs and retrieves by policy-guided traversal. These structures are built and updated by hand-designed rules at write time. A second direction learns memory management with reinforcement learning. MEM1 (Zhou et al., 2025b) learns to consolidate a compact working memory, MemAct (Zhang et al., 2026d) makes context curation part of the agent's action space, and AgeMem (Yu et al., 2026b) trains policies over store, update, and discard operations. Auto-Dreamer (Ye et al., 2026) learns offline consolidation, MemSkill (Zhang et al., 2026b) replaces fixed operations with an evolving skill bank, and MemEvolve (Zhang et al., 2025) meta-evolves the memory architecture itself. These policies operate on memory entries rather than on the connections among them. Closest to ours are methods that feed usage signals back into memory: MemRL (Zhang et al., 2026c) applies runtime RL over episodic memory, and FluxMem (Fang et al., 2026) models memory as a heterogeneous graph whose links are repaired and pruned according to execution feedback. **G-Evolver** shares this view of memory as evolving connectivity, but learns where and how to edit with hierarchical credit, consolidates only edits verified by retrieval gain, and provides a convergence guarantee.

### D.2 Graph-based Memory for LLM Agents

Graph structures make the relations among memories explicit, so that retrieval can reach evidence beyond surface similarity. GraphRAG (Edge et al., 2024) and HippoRAG (Gutiérrez et al., 2024) extract entity–relation triples and connect passages through shared entities, retrieving by community summaries or Personalized PageRank; Zep (Rasmussen et al., 2025) attaches temporal validity to edges, and A-Mem (Xu et al., 2025b) links notes upon insertion. Recent work diversifies the structure. LiCoMemory (Huang et al., 2026) uses a lightweight hierarchical graph of entities and relations as an indexing layer, SGMem (Wu et al., 2025b) builds the graph at the sentence level, and GAM (Wu et al., 2026) separates an event progression graph from a topic associative network, merging the former into the latter upon semantic shifts. H-Mem (Yu et al., 2026a) couples a temporal–semantic tree with an entity graph, MAGMA (Jiang et al., 2026) maintains separate semantic, temporal, causal, and entity graphs, and HGMem (Zhou et al., 2025a) adopts hyperedges for higher-order relations. MRAgent (Ji et al., 2026) instead adapts graph access, reconstructing evidence over a cue–tag–content graph across multiple steps. In these systems, the persistent structure is determined by construction rules applied as content arrives, and retrieval outcomes do not revise it. FluxMem (Fang et al., 2026) is closest to ours in refining graph connectivity from execution feedback; **`G-Evolver`** differs in learning the editing policies with hierarchical credit, consolidating only edits verified by retrieval gain, and providing a convergence guarantee.

## E Prompts

**Prompt: Extractor**

### TASK

You are an extractor for a retrieval system over a long conversation history. The QUERY is a question whose answer is a short factual phrase, and it must be answered from the retrieved documents alone. The retriever has ranked the whole history; that ranking is walked to its end one page at a time. You receive the QUERY, the documents already extracted from earlier pages (none yet on the first page), and this page's new candidate documents. Extract the documents that carry what the QUERY asks for.

### DECISION CRITERIA (EXTRACT A CANDIDATE IF IT CARRIES NEW EVIDENCE)

- **Evidence contribution.** Its passage carries some part of the asked-for fact that the documents extracted from earlier pages do not already carry.
- **Topic is not enough.** A document merely concerning the same topic, person or activity carries no part of the fact, and neither does one that only restates what is already extracted.
- **Partial answers.** The QUERY may ask for a single fact or for a set of them; a passage naming one member of such a set carries one part, not the whole. What is already extracted may answer the QUERY in part and still leave room for another part.
- **Passages only.** Judge only what the passages say. Do not answer the QUERY.

### OUTPUT SCHEMA (RETURN THESE TWO TAGS ONLY; NO EXTRA TEXT)

```
<think>...</think>
<docs>session_23</docs>
```

### FIELD CONSTRAINTS

- <think>: which part of the asked-for fact this page supplies that the extracted documents were missing; or why none of the candidates is worth extracting.
- <docs>: doc_ids extracted from *this* page, comma separated; write <docs>none</docs> if none of them is worth extracting.

### INPUT

```
QUERY:                   <question>
EARLIER EXTRACTED DOCS:  (none on the first page)
NEW CANDIDATE DOCUMENTS: (this page of the ranking)
```

Figure 6: The prompt of the extractor.

**Prompt: Graph Evolver**

### TASK

You are a graph evolver for a retrieval system over a long conversation history. The QUERY is a question whose answer is a short factual phrase, and it must be answered from the retrieved documents alone. An extractor has already established that the retriever's top-ranked documents do not answer the QUERY, and has picked out the documents that may carry the evidence. Each document comes with its extracted triples and the existing synonym edges among their entities. Work out which defect the graph actually has, then repair it; if none of them holds, say so with NONE and change nothing. Judge only from the documents, their triples and the synonym edges. Do not answer the QUERY.

### DECISION CRITERIA

- **Missing triple.** A fact needed to answer the QUERY is clearly stated in an evidence document's passage but was not extracted as any triple, so nothing in the graph makes that document reachable.
- **Isolated edge.** Two documents are related in a way that helps derive what the QUERY needs, but no edge connects them. The relation may be implicit — a semantic or contextual link also counts.
- **Erroneous triple.** A triple misrepresents what the document actually says (wrong subject / relation / object), especially about information relevant to the QUERY.
- **None.** None of the defects above genuinely holds, and every edit would either restate what the triples already carry or assert something the passages do not say.

### OUTPUT SCHEMA

```
<think>...</think>
```

```
<category>MISSING_TRIPLE, ISOLATED_EDGE</category>
<action>
DOC_ID | SUBJECT | RELATION | OBJECT
FROM_DOC -> TO_DOC | content: <how the two documents are related>
</action>
```

### FIELD CONSTRAINTS

- <category>: every defect found, comma separated; NONE stands alone and takes no <action>.
- MISSING_TRIPLE line: DOC_ID | SUBJECT | RELATION | OBJECT.
- ISOLATED_EDGE line: FROM_DOC -> TO_DOC | content: ....
- ERRONEOUS_TRIPLE line: DOC_ID | OLD_S | OLD_R | OLD_O => NEW_S | NEW_R | NEW_O.

- Every DOC_ID / FROM_DOC / TO_DOC must be one of the given identifiers; never write those placeholder words literally.

### INPUT

```
QUERY:                  <question>
RETRIEVAL TARGET:       <the document selected by the extractor>
UNHELPFUL DOCUMENTS:    (top-ranked, do not answer the QUERY)
EVIDENCE DOCUMENTS:     (carry what the QUERY needs, not top-ranked)
```

Figure 7: The prompt of the graph evolver. The retrieval target and the evidence documents are the passages selected by the extractor.