EVOMEM: TYPE-ADAPTIVE RETRIEVAL OVER VERSIONED MEMORY FOR SELF-EVOLVING LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term memory enables LLM agents to retain user-specific information across extended interactions. However, maintaining evolving user attributes requires identifying their current values and their historical states at specified times. Existing approaches either retrieve outdated values from an insufficiently updated knowledge base or compress observations into summaries that obscure revision histories. Moreover, most systems tightly couple memory storage and retrieval, thereby imposing a fixed access path regardless of query requirements. To address these limitations, we introduce EvoMem, a memory architecture that supports both current-state and historical reasoning by decoupling versioned storage from query-adaptive retrieval. EvoMem preserves complete histories of user attributes without committing them to a single retrieval structure, and dynamically selects and ranks evidence according to the query type. This design poses two key challenges. Different queries require distinct retrieval mechanisms, while retaining all attribute versions may cause obsolete information to compete with current values and lead to incorrect retrieval. EvoMem addresses these challenges through type-adaptive retrieval and self-evolving memory consolidation. Type-adaptive retrieval selects appropriate memory structures and retrieval paths for factual, time-sensitive, and multi-session reasoning queries. Self-evolving memory consolidation integrates Versioned Knowledge Evolution (VKE), which preserves timestamped attribute versions, with Error-Driven Pattern Learning (EDPL), which extracts reusable patterns from prior retrieval errors to continually refine ranking policies. Experiments on LongMemEval and LoCoMo demonstrate that EvoMem consistently outperforms strong baselines under predicted query-type routing. In particular, on LoCoMo, EvoMem exceeds the strongest baseline by 3.6 points in QA accuracy and 7.1 points in token-level F1. The source code is available at https://anonymous.4open.science/r/Evomem/.

1 INTRODUCTION

Long-term memory is fundamental to LLM agents deployed in persistent, real-world settings (Hu et al., 2026). For example, a personal assistant should recognize when a user's preference changes from coffee to tea and accordingly revise its future recommendations. Similarly, a collaborative task-management agent must retain decisions and contextual information across long-running projects (Yao et al., 2023; Qian et al., 2024). In both settings, effective agents must not only retrieve previously observed information but also track and reason over the temporal evolution of user attributes, preferences, and decisions (Liu et al., 2024).

However, contemporary LLMs process interactions within finite context windows and typically lack persistent state across sessions (Achiam et al., 2023; Shi et al., 2023). Consequently, an agent may fail to maintain consistency with decisions made in earlier interactions (Li et al., 2025). Although external memory systems introduce persistent storage and retrieval mechanisms (Graves et al., 2014; Behrouz et al., 2026), their ability to represent evolving attributes remains limited. Retrieval-augmented approaches (Gao et al., 2023; Lewis et al., 2020) often preserve interaction histories without explicitly representing when previously valid values become obsolete. Conversely, memory-augmented systems (Cai et al., 2025; Packer et al., 2023) progressively consolidate observations, potentially obscuring the temporal trajectory and revision history of changing attributes. As illustrated in Figure 1, a further limitation is that existing systems commonly couple memory storage

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.

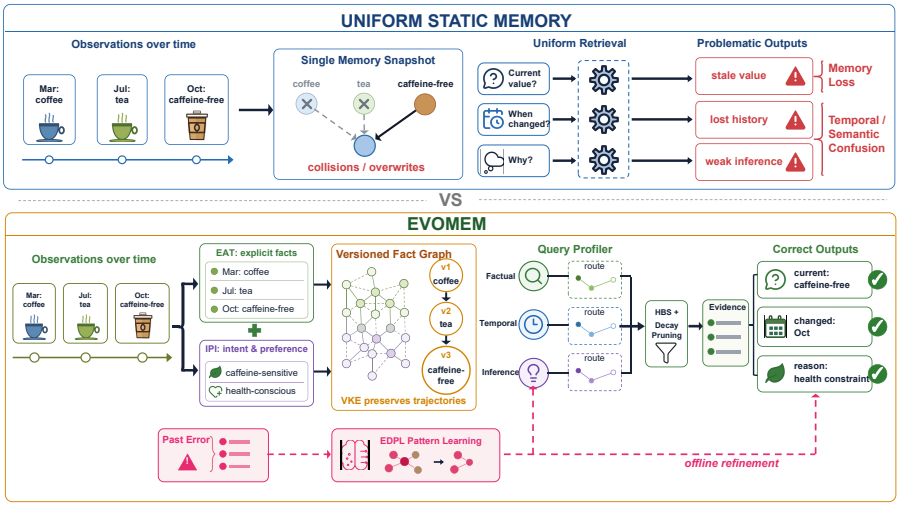

Figure 1: Comparison of prior memory-augmented systems and EvoMem.

and retrieval within a unified workflow. As a result, the stored representation simultaneously determines the memory structures searched and the retrieval procedure applied to every query (Chhikara et al., 2025; Xu et al., 2026b). This coupling prevents current-state, historical-state, and multi-session reasoning queries from independently selecting the retrieval components and access paths required to obtain appropriate evidence from a shared, evolving interaction history.

To address these limitations, we propose **EvoMem**, a memory architecture that decouples versioned memory storage from query-adaptive retrieval. EvoMem maintains complete revision histories for user attributes without committing them to a single retrieval structure or access path. This design supports queries concerning an attribute's current state, its value at a specified historical point, and its temporal evolution. For example, EvoMem can identify a user's current city of residence, recover the city recorded at an earlier time, and determine when the corresponding transition occurred.

However, two technical challenges remain. First, distinct query types require different forms of evidence from the versioned history of the same attribute. For example, identifying a user's current beverage preference requires prioritizing the most recent valid observation, whereas recovering the preference at a specified time requires locating the relevant state transition. A single, fixed retrieval workflow is therefore inadequate, since applying an identical procedure to all attribute versions may fail to retrieve the transition event or cross-session context required by a given query. Second, the system must preserve complete attribute histories without allowing obsolete values to interfere with current-state retrieval. Retaining every version can cause superseded values to compete with the latest valid state, while overly aggressive pruning may discard the evidence necessary to reconstruct an attribute's temporal evolution (Dai et al., 2026a).

EvoMem addresses the first challenge through type-adaptive retrieval, which maps each query to a retrieval profile defined over factual, temporal, and inferential dimensions. Based on this profile, EvoMem activates the corresponding memory components and retrieval paths, and ranks candidate evidence accordingly. Specifically, factual queries retrieve explicit entity–attribute facts; temporal queries traverse attribute-specific version chains to recover historical states and transitions; and multi-session reasoning queries combine fact-graph paths with the source episodes that establish relevant cross-session context (Edge et al., 2024; Sarthi et al., 2024).

To address the second challenge, EvoMem employs self-evolving memory consolidation, which integrates Versioned Knowledge Evolution (VKE) with Error-Driven Pattern Learning (EDPL). Rather than overwriting previous values, VKE appends each observation as a timestamped node to an attribute-specific version chain, thereby preserving the complete revision history. EDPL analyzes feedback-identified retrieval failures to refine the retrieval and ranking policies. Subsequently, time-aware ranking and query-adaptive pruning prioritize evidence relevant to the current query without deleting historical records. EDPL is learned from development feedback and frozen before evaluation; in deployment, policy updates are performed only after user feedback becomes available. Thus,

the term self-evolving denotes feedback-driven adaptation of retrieval policies over time, whereas updates to the versioned memory store are performed by VKE at ingestion.

Our contributions are summarized as follows:

- We propose EvoMem, a memory architecture that decouples versioned storage from query-adaptive retrieval, enabling LLM agents to track evolving user facts and preferences across sessions.
- We introduce type-adaptive retrieval, which profiles queries along factual, temporal, and inference dimensions and activates matching memory structures and retrieval strategies.
- We introduce self-evolving memory consolidation, which combines VKE to preserve timestamped attribute trajectories with EDPL to learn reusable retrieval and ranking guidance from past feedback errors.
- On LoCoMo and LongMemEval (Maharana et al., 2024; Wu et al., 2025), EvoMem with predicted query types achieves 80.0/38.6 and 85.2/39.6 in accuracy/token-level F1, respectively. Gold query-type routing provides an oracle upper bound of 82.6/42.5 and 88.4/44.5.

2 Related Work

Structured Memory Systems. Retrieval-augmented generation Gao et al. (2023); Lewis et al. (2020) typically retrieves evidence from a static corpus that is not explicitly updated by subsequent user interactions. Consequently, outdated mentions may remain retrievable. Entry-based memory systems instead represent conversational histories as flat entries and consolidate recurring observations, as MemoryBank Zhong et al. (2024) and RecMem Dai et al. (2026b) do. More structured approaches encode memories as nodes and edges, supporting tree- or graph-based retrieval Edge et al. (2024); Sarthi et al. (2024) and semantic, temporal, causal, or entity-centric graphs Jiang et al. (2026). Mnemis Tang et al. (2026) combines embedding search with traversal over a hierarchical concept graph, whereas ARTEM Tan et al. (2026a) retrieves through multi-channel vigilance. Recent work organizes such structures hierarchically Wu et al. (2026); Xu et al. (2026a) and extends them to user profiles Huang et al. (2026); Alonso et al. (2024). Although Mem0 Chhikara et al. (2025) and some temporal knowledge-graph stores maintain timestamped memories, they do not model per-attribute version chains or select retrieval strategies per query, so existing designs emphasize episodic and event-level relations rather than the temporal evolution of user attributes.

Self-Evolving Memory. Self-evolving memory systems improve agent behavior by accumulating and reusing experience across interactions Shinn et al. (2023); Wang et al. (2023); Park et al. (2023). Some approaches distill experience into procedural memory revised as the agent acquires new capabilities Cao et al. (2026). Evo-Memory Wei et al. (2025) introduces an action–think–memory pipeline for continual self-improvement, whereas MUSE-Autoskill Fu et al. (2026) accumulates skill-specific experience under a unified memory lifecycle. Experience-driven lifelong learning formulates memory evolution as a measurable process with benchmarks Cai et al. (2025). MemRL Zhang et al. (2026) and ExpeL Zhao et al. (2024) improve policies through reinforcement learning and experience replay. Memory may also evolve during inference since test-time memorization enables models to acquire, store, and recall information without a separate training phase Behrouz et al. (2026). These approaches target procedural or task-level experience but do not track fine-grained user attributes, including their revisions, across sessions.

3 Methodology

Overview. Figure 2 presents an overview of EvoMem, which decouples versioned memory storage from query-adaptive retrieval through three coordinated components. Section 3.1 describes the ingestion module, which constructs four complementary memory structures from each completed interaction session: EAT, FG, FEI, and IPI. Detailed construction procedures are provided in Appendix A.6. Section 3.2 introduces type-adaptive retrieval to address the first challenge: it characterizes each query along factual, temporal, and inferential dimensions and activates the structures and paths aligned with the resulting profile. Section 3.3 addresses the second challenge through self-evolving consolidation: VKE preserves the version chain of each attribute, while EDPL leverages historical feedback to refine retrieval and ranking. Finally, Section 3.4 presents the complexity analysis of the overall design, with detailed derivations deferred to Appendix A.5.

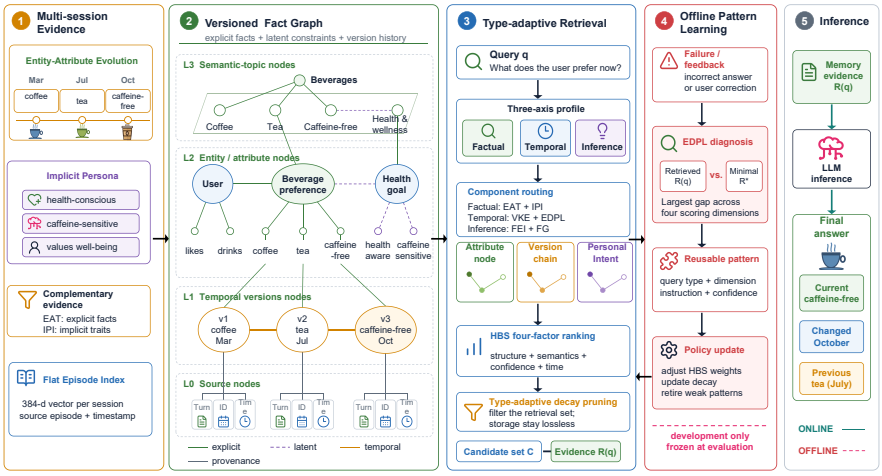

Figure 2: The EvoMem architecture. Offline ingestion constructs Entity-Attribute Tracking (EAT), the Fact Graph (FG), the Flat Episode Index (FEI), and Intent and Preference Inference (IPI). Query-adaptive retrieval uses query profiling, Heuristic Beam Search (HBS), and pruning. Versioned Knowledge Evolution (VKE) preserves attribute histories, and Error-Driven Pattern Learning (EDPL) uses feedback to improve later retrieval.

3.1 Memory Ingestion

Each interaction session contains multiple complementary forms of information, including explicit facts, full conversational context, and task-relevant preferences or decisions. EvoMem stores these information types in separate but linked memory structures, enabling query-adaptive retrieval to access the most relevant evidence while preserving the complete interaction history for subsequent reasoning.

Entity-Attribute Tracking (EAT). EAT extracts facts that are explicitly stated by users. It employs a fixed library of syntactic templates for actions, attributes, quantities, persons, and frequencies. Given an interaction session S_i, EAT applies the template set Π to each conversational turn and constructs the associated fact set as

$$F(S_i) = \bigcup_{t \in \text{turns}(S_i)} \text{match}(t, \Pi). \tag{1}$$

Each extracted fact $f \in (F(S_i))$ is represented as $(f = (s, p, o_v, t, c, mathrmsrc))$, where s, p, o_v, t, c, and src denote the subject, predicate, value, timestamp, extraction confidence, and source session, respectively. EAT records only explicitly expressed information and does not infer unstated attributes. Each fact stays linked to its originating session, so later queries can access either the fact or its conversational context. These records establish what was stated in each session, but a flat fact collection cannot capture the temporal evolution of an attribute. The Fact Graph (FG) therefore organizes these facts into temporally structured attribute histories.

Fact Graph (FG). FG makes entity-attribute evolution queryable by organizing EAT facts into a four-layer graph, $G_{\text{FG}} = (V, E)$: semantic topic (L3), entity and attribute (L2), temporal version (L1), and source (L0) nodes. Topic nodes group related attributes, and entity and attribute nodes come from the subjects, speakers, and predicates of extracted facts. Each observation creates a temporal version node with its value and timestamp and each source node records the session and turn of the observation. The graph encodes three relations: (1) entities are linked to their attributes, (2) consecutive observations of the same attribute are connected in chronological order, and (3) each version node is linked to the provenance node supporting it. This structure enables temporally grounded queries to resolve distinct states within the same attribute history. Although FG captures the evolution of structured attributes, certain queries require the original wording and broader context of an entire interaction session. Such session-level evidence is preserved by FEI..

Flat Episode Index (FEI). FEI preserves each completed session as a retrievable episode. For a session S_i, it computes an embedding $e_i = \text{emb}(S_i)$ Reimers & Gurevych (2019) and stores $I_{\text{base}} = \{e_1, \ldots, e_N\}$ together with the session text and timestamp. Given a query embedding e_q,

FEI returns the top-k with the highest cosine similarity to e_q. While full-session retrieval preserves context and provenance, similarity over episode embeddings alone may not reliably identify recurring preferences or task constraints. IPI provides this complementary pattern-level representation while maintaining links to the original sessions.

Intent and Preference Inference (IPI). IPI uses a single LLM call per session to extract explicitly stated, task-relevant user information, including preferences, goals, task decisions, and interaction constraints. Each profile entry retains its associated timestamp and source session. These explicit fields are also represented in FG as timestamped attributes, enabling their values to form version chains across sessions. For queries concerning user preferences, goals, or task decisions, IPI retrieves the corresponding profile entries as supplementary context. The complete field schema and prompt template are provided in Appendix A.7. Collectively, EAT, FG, FEI, and IPI preserve information at the levels of facts, temporal versions, complete sessions, and explicit user profiles.

3.2 Type-Adaptive Retrieval

Different query types require distinct forms of evidence. For example, determining a user's current preference requires the most recent supported observation, whereas recovering a prior preference requires identifying the relevant transition in its version chain Tan et al. (2026b). EvoMem therefore selects a retrieval path aligned with the query's evidence requirements rather than applying one fixed procedure to all queries Asai et al. (2024); Du et al. (2026).

Query profiling and routing. At query time, EvoMem uses a lightweight router to assign each query a type T_q (Appendix B.6). A fixed table maps each query category to the profile

$$p(T_q) = (f_s, t_s, d_i), \tag{2}$$

where the binary components f_s, t_s, and d_i of Eq. 2 indicate the need for explicit facts, temporal information, and reasoning across sessions, respectively. By default, FEI and FG provide the starting point for every query. In addition, explicit-fact queries also consult EAT and IPI, and temporal queries follow the relevant version chain in FG.

Heuristic Beam Search (HBS). The selected memory structures first construct a candidate set $C(q)$. FEI contributes similar sessions, EAT matching explicit facts, and IPI relevant profile entries. HBS then refines the FG-derived candidates without traversing the whole graph: it aligns entity mentions with entity nodes, identifies their attribute nodes, and starts from the associated version chains. At each expansion it retains the b highest-scored paths, where b is the beam width.

For a candidate version n_j reached from n_i, the HBS score is

$$S(n_j \mid n_i, q) = \alpha_{T_q} \Phi_{\text{struct}}(e_{ij}, T_q) + \beta_{T_q} \operatorname{sim}(\vec{n}_j, \vec{q}) + \gamma_{T_q} c_{\text{cal}}(n_j) + \delta_{T_q} f_{\text{rec}}(n_j, t_q). \tag{3}$$

The scoring function S combines four signals: (1) Structural alignment $\Phi_{\text{struct}}(e_{ij}, T_q) = w_{T_q}^{\top} \phi(e_{ij})$ scores the edge type of each transition, where ϕ encodes the four edge types of Appendix A.6 and w_{T_q} is a type-specific weight vector. (2) Semantic relevance $\operatorname{sim}(\vec{n}_j, \vec{q})$ is the cosine similarity between the node and query embeddings. (3) Calibrated confidence $c_{\text{cal}}(n_j) = (\kappa + s_c)/(\kappa + m_c)$ summarizes the extraction reliability of a version node, where each observation contributes its stated confidence (low, medium, high $\to 0.3, 0.7, 1.0$), s_c is their sum, m_c the number of observations, and $\kappa = 3$ a smoothing pseudo-count. EDPL tracks a different quantity, the success rate of a learned pattern, with a unit-prior Beta-Binomial posterior mean in Appendix A.4. (4) Temporal freshness $f_{\text{rec}}(n_j, t_q) = 1/(1 + \eta_{T_q} |t_q - t_j|)$ down-weights nodes far from the query time, where t_j is the timestamp of n_j and η_{T_q} is a type-adaptive rate. The parameters $\theta_T = (\alpha_T, \beta_T, \gamma_T, \delta_T, w_T)$ weight these signals for query type T, with α_T for structural alignment, β_T semantic similarity, γ_T calibrated confidence, δ_T temporal freshness, and w_T the edge-type weights. On the development split $\mathcal{D}_{\text{dev}}$, each query q has a minimum support set $R^*(q)$ of provenance-linked version nodes from which the reference answer can be derived. HBS learns θ_T by scoring $R^*(q)$ above the remaining candidates $C(q)$ with a support-evidence ranking loss:

$$\mathcal{L}_{\text{HBS}} = - \sum_{(q, T_q) \in \mathcal{D}_{\text{dev}}} \log \frac{\sum_{n \in R^*(q)} \exp S(n \mid \text{pa}(n), q)}{\sum_{n \in C(q)} \exp S(n \mid \text{pa}(n), q)} + \lambda_{\text{reg}} \sum_{T} \|\theta_T - \theta_0\|_2^2, \tag{4}$$

where $\text{pa}(n)$ is the parent of n, θ_0 a shared parameter vector, and λ_{reg} controls the regularizer strength. HBS retains the strongest candidates, but a highly ranked version can still be outdated or weakly supported; query-adaptive pruning performs the final filtering step.

Query-adaptive pruning. After HBS ranks the candidates, EvoMem filters outdated or weakly supported version nodes without deleting them from FG. A candidate version n_j keep score as follows:

$$\text{keep}(n_j) = f_{\text{conf}}(c) \cdot f_{\text{src}}(w) \cdot \frac{1}{1+\eta\Delta t} \cdot \frac{1}{1+\lambda\Delta t_{\text{access}}}, \tag{5}$$

where c is the extraction confidence of the candidate and w the session support accumulated from the observations that report it, f_{conf} maps c to the discrete levels $\{0.3, 0.7, 1.0\}$, and $f_{\text{src}}(w) = 1 - e^{-w}$ rewards support from multiple sessions. In the third term, Δt is the interval between the query time and the timestamp of the candidate and η is its decay rate. In the final term, Δt_{access} is the elapsed time since the candidate was last retrieved and λ gives a small preference to recently accessed nodes and reduces repeated retrieval of stale evidence. Finally, a pruning threshold $\theta^{\text{prune}}_{T_q}$ selected for query type T_q determines which candidates remain in the retrieved set, keeping complete histories in storage while preventing stale or weakly supported evidence from dominating the answer.

3.3 Self-Evolving Memory

EvoMem must preserve historical attribute values while improving retrieval when similar errors recur. Its self-evolving consolidation addresses both objectives: VKE retains each observed value as a timestamped version in FG, preserving complete attribute histories, whereas EDPL uses feedback-identified errors to refine later retrieval and ranking.

Versioned Knowledge Evolution (VKE). VKE updates attribute records while preserving their complete historical trajectories. For each new observation, it appends a temporal version node to the version chain associated with the corresponding entity–attribute pair in FG. If an identical value is observed on the same date, VKE increases the support of the existing version node by 0.5, rather than creating a duplicate node. Each new node is linked to its preceding version, explicitly encoding temporal transitions. Because HBS and pruning operate only over retrieved candidates, VKE never removes stored versions, so FG retains the evidence required to resolve current values, historical values, and transitions.

Error-Driven Pattern Learning (EDPL). When feedback indicates an incorrect answer, EDPL analyzes the completed interaction to derive a candidate correction pattern. It then performs a counterfactual diagnosis by contrasting the retrieved evidence with the minimal evidence required to support the corrected answer. This analysis attributes the failure to one or more sources, including graph structure, semantic matching, extraction confidence, and temporal relevance. EDPL updates the retrieval policy only when repeated diagnoses increase the pattern confidence beyond a predefined activation threshold. Each diagnosis produces a candidate pattern $p = (\tau_p, \iota_p, c_p)$, namely trigger terms or query conditions τ_p, a corrective retrieval instruction ι_p, and confidence c_p. EDPL groups candidate patterns with the same query conditions and retrieval cause, then updates their confidence as feedback accumulates.

3.4 Complexity Analysis

Versioned storage grows linearly with the number of observations. FEI stores N vectors of dimension d, requiring $O(Nd)$; EAT requires $O(|F|)$; and FG requires $O(|V_{\text{attr}}| + M)$ for attribute nodes and M observations. At query time, FEI search costs $O(Nd)$, and HBS costs $O(bLd)$ for each matched entity, where L is the version-chain length and b is the beam width. IPI uses one API call per session, and EDPL uses one diagnosis call per interaction that receives an error signal, respectively. Overall, storage stays linear in the number of observations and graph search is restricted to a beam over the chains of matched attributes. Appendix A.5 gives the full derivation.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate on two long-term conversational memory benchmarks: LoCoMo and LongMemEval (Maharana et al., 2024; Wu et al., 2025). LoCoMo comprises multi-session dialogues averaging roughly 300 turns and 9K tokens each. Its questions are categorized as multi-hop, temporal, open-domain, single-hop, and adversarial; following prior work, we exclude the adversarial

category. LongMemEval includes six categories: single-session user (SSU), single-session assistant (SSA), single-session preference (SSP), temporal reasoning (TR), knowledge update (KU), and multi-session (MS). For each benchmark we evaluate on a 500-question subset, using 100 questions as a development split for EDPL training and the remaining 400 for testing.

Baselines and evaluation protocol. We compare EvoMem against BM25 RAG and five memory systems: MemoryBank Zhong et al. (2024), PREMem Kim et al. (2025), Mnemis Tang et al. (2026), RecMem Dai et al. (2026b), and LightMem Fang et al. (2026). Our primary setting uses *predicted routing*, in which a lightweight router infers the query type from the query; we also report *gold routing*, which uses benchmark-provided query types as an oracle upper bound. For LongMemEval, we employ a zero-shot LLM-based router that classifies queries using surface-level cues and achieves 91.6% accuracy across the six query types. For LoCoMo, a logistic-regression classifier over `bge-small-en-v1.5` embeddings maps each query to its benchmark-defined type. The development split is disjoint from the test set and supports EDPL pattern learning, HBS weight tuning, and pruning-threshold calibration.

Table 1: Overall and per-category results (accuracy / token-level F1, %) on the test splits of LongMemEval and LoCoMo. All methods use the DeepSeek-V4-Flash backbone; EvoMem (gold) routes with query types annotated by the benchmarks, EvoMem (predicted) with types inferred by a classifier.

	LongMemEval						
Method	SSU	SSA	SSP	TR	KU	MS	Overall
BM25 RAG	60.0 / 24.6	90.0 / 32.3	59.0 / 23.8	55.0 / 23.1	50.0 / 23.1	75.0 / 27.8	64.8 / 25.8
MemoryBank	85.0 / 32.3	88.0 / 37.0	79.0 / 31.5	40.0 / 25.4	80.0 / 35.5	50.0 / 27.8	70.4 / 31.6
PREMem	76.0 / 34.0	68.0 / 26.3	75.0 / 33.5	72.0 / 31.6	84.0 / 38.5	78.0 / 35.5	75.4 / 33.2
Mnemis	92.0 / 41.7	100.0 / 43.2	79.0 / 36.5	80.0 / 37.0	70.0 / 35.5	78.0 / 35.5	83.2 / 38.2
RecMem	85.0 / 37.0	100.0 / 40.1	73.0 / 36.0	75.0 / 34.0	74.0 / 38.5	61.0 / 27.8	78.0 / 35.6
LightMem	70.0 / 40.0	72.0 / 42.0	69.0 / 36.5	70.0 / 37.0	66.0 / 34.0	65.6 / 32.4	68.8 / 37.0
EvoMem (gold)	88.0 / 47.6	100.0 / 50.9	82.2 / 43.4	91.2 / 43.0	87.0 / 38.6	82.0 / 43.5	**88.4 / 44.5**
EvoMem (predicted)	87.0 / 46.0	100.0 / 48.0	78.8 / 38.5	89.0 / 39.5	84.0 / 34.0	72.4 / 31.6	**85.2 / 39.6**

	LoCoMo				
Method	Multi-hop	Temporal	Open-domain	Single-hop	Overall
BM25 RAG	59.0 / 20.1	59.0 / 22.1	57.0 / 19.5	67.0 / 25.5	60.8 / 22.1
MemoryBank	65.0 / 24.2	70.4 / 26.1	66.0 / 24.2	72.0 / 27.5	68.4 / 25.5
PREMem	67.4 / 24.8	70.4 / 26.8	73.0 / 27.5	74.4 / 28.9	70.8 / 26.8
Mnemis	71.4 / 30.2	77.0 / 31.5	75.0 / 30.8	79.0 / 32.2	76.4 / 31.5
RecMem	68.4 / 26.8	74.0 / 28.9	72.0 / 28.2	78.0 / 30.8	73.6 / 28.9
LightMem	70.0 / 31.8	75.0 / 38.5	71.0 / 36.5	72.0 / 40.4	71.6 / 36.8
EvoMem (gold)	80.0 / 33.6	83.4 / 43.4	80.0 / 36.0	86.6 / 57.0	**82.6 / 42.5**
EvoMem (predicted)	84.0 / 37.4	88.0 / 39.0	68.0 / 36.6	89.0 / 44.0	**80.0 / 38.6**

Metrics. We report LLM-as-a-Judge accuracy Zheng et al. (2024) and token-level F1 Chen et al. (2025), aggregated overall and by question category. LLM-based evaluations use an independent GLM-5.3-Flash model, separate from the generator. Appendix B provides detailed descriptions of the datasets, baselines, implementation settings, evaluation metrics, and routing protocol.

4.2 Overall Results

Table 1 summarizes overall and per-category results; predicted routing is the deployment setting and gold routing quantifies the upper bound.

LongMemEval. EvoMem attains the best overall accuracy and F1 on LongMemEval under both settings, reaching 85.2% accuracy and 39.6% F1 with predicted routing, and 88.4% and 44.5% with gold routing, In terms of overall accuracy, EvoMem surpasses Mnemis, the strongest baseline, by 2.0% and 5.2% accuracy points under predicted and gold routing, respectively. Its largest gains arise in categories that require reasoning over evolving memory states. Under gold routing, EvoMem achieves 91.2% accuracy on temporal-reasoning and 82.0% on multi-session questions, exceeding the best baseline by 11.2% and 4.0%, respectively. Its largest advantage over Mnemis is on knowledge-update questions (87.0% against 70.0%). These results indicate that versioned evidence tracking is particularly effective for queries whose correct answers depend on evolving or historically contingent user states, and that these benefits are largely retained under predicted routing.

LoCoMo. EvoMem attains the best accuracy and F1 on LoCoMo under both settings: 80.0% and 38.6% with predicted routing, and 82.6% and 42.5% with gold routing, exceeding Mnemis, the strongest accuracy baseline, by 3.6 and 6.2 points, respectively. It also surpasses LightMem, the strongest F1 baseline, by 5.7 F1 points.

The improvements are particularly pronounced for question types that require extracting or aggregating evidence across the dialogue history. Under gold routing, EvoMem achieves 86.6% accuracy on single-hop and 83.4% on temporal questions, outperforming the best baseline by 7.6 and 6.4 points. Its largest margin over Mnemis is on multi-hop questions (80.0% versus 71.4%). These results suggest that versioned evidence tracking effectively supports reasoning over long, multi-session dialogues, with consistent accuracy improvements across question types.

Table 2: Ablation study (accuracy / F1, %) on LoCoMo and LongMemEval.

	LoCoMo		LongMemEval	
Ablation	Acc	F1	Acc	F1
EvoMem	**82.6**	**42.5**	**88.4**	**44.5**
Memory Ingestion				
w/o EAT	68.2	25.9	74.6	33.3
w/o IPI	72.0	28.0	73.8	32.0
w/o FG	75.6	29.8	76.2	35.2
Type-Adaptive Retrieval				
w/o Profiling	64.6	23.3	67.4	28.7
w/o HBS	72.8	27.1	73.2	31.7
Self-Evolving Memory				
w/o EDPL	66.4	23.5	68.2	29.2
w/o VKE	70.4	26.7	72.6	31.6

Table 3: Routing strategy comparison (accuracy, %) on LoCoMo and LongMemEval.

	Accuracy (%)	
Routing	LoCoMo	LongMemEval
gold (oracle)	**82.6**	**88.4**
predicted	80.0	85.2
uniform	70.0	75.0
soft	60.0	62.4
random	62.4	57.4
shuffled	60.0	45.0

Table 4: Accuracy (%) on the LongMemEval slice of changing attribute values, contrasting EvoMem with a FullText-30K baseline.

Method	Current	Historical	Conflict	Overall
EvoMem	**76.6**	**88.2**	**81.2**	**82.0**
FullText-30K	47.2	64.8	68.8	60.2

4.3 Ablation and Analysis

Component ablations. Table 2 evaluates the contribution of EvoMem's major components. Removing query profiling yields the largest degradation, reducing accuracy from 82.6% to 64.6% on LoCoMo and from 88.4% to 67.4% on LongMemEval. This result highlights the importance of query-adaptive routing: it identifies the evidence sources whose outputs are subsequently scored and integrated by the remaining modules. Without this routing mechanism, the effectiveness of downstream retrieval and reasoning components is substantially diminished.

Removing HBS reduces accuracy to 72.8% and 73.2%, removing EDPL to 66.4% and 68.2%, and removing VKE to 70.4% and 72.6%. Ablations of the ingestion modules produce comparatively smaller, yet still substantial, degradations: removing EAT yields 68.2% and 74.6%, removing IPI 72.0% and 73.8%, and removing FG 75.6% and 76.2% on LoCoMo and LongMemEval, with FG producing the smallest degradation of all. These results show that explicit fact extraction, version-preserving representations, and complete source-session information provide complementary evidence for long-horizon reasoning.

Effect of routing quality. Table 3 isolates routing quality. Gold to predicted routing costs only 2.6 accuracy points on LoCoMo and 3.2 on LongMemEval, so the router learns a usable approximation of the oracle decision. Uniform routing reaches only 70.0% and 75.0%, and random and shuffled routing fall below it, These results demonstrate that In contrast, uniform routing achieves only 70.0% accuracy on LoCoMo and 75.0% on LongMemEval, while random and shuffled routing perform worse than the uniform routing control. These results demonstrate that EvoMem's gains stem from routing quality rather than from the availability of multiple retrieval branches.

The *w/o Profiling* condition differs from uniform routing, which weights all structures equally: removing the query-conditioned activation policy lowers accuracy further to 64.6% and 67.4%. EvoMem also narrows the per-category accuracy range on LongMemEval from 40 points for BM25 RAG to 18 under gold routing; additional routing controls appear in Appendix B.6.

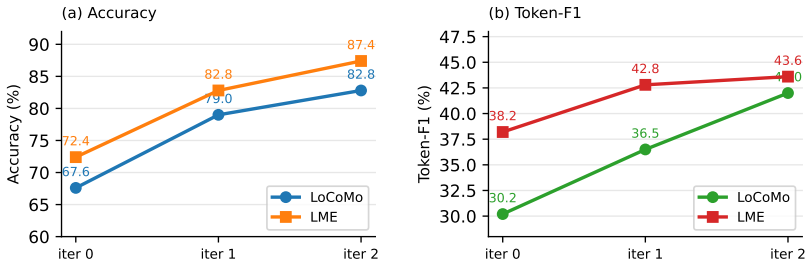

Figure 3: EDPL learning curves on the development split of LoCoMo and LongMemEval under gold routing. Accuracy (a) and token-level F1 (b) improve with each feedback iteration.

Versioned tracking under a fixed context budget. Conventional memory systems retain only the most recent statement about an attribute, which makes changing values hard to track, whereas EvoMem preserves the full sequence of versions. To disentangle version-aware tracking from increased context capacity, we compare EvoMem with a FullText-30K baseline on 17 changing attributes under the same 30,000-character budget. As Table 4 reports, EvoMem reaches 82.0% accuracy against 60.2% for FullText-30K. The largest gain occurs on current-value questions, where EvoMem exceeds FullText-30K by 29.4 points, substantial on historical questions, exceeding the baseline by 23.4 points because long context retains earlier statements, and smallest on conflict questions, where the gap narrows to 12.4 points. Overall, these results indicate that EvoMem's advantages on dynamically changing attributes arise primarily from explicit versioned attribute tracking, rather than from access to a larger context window.

Cost. We measure per-question API usage on an independent sample of 100 questions per benchmark. Most EvoMem calls come from offline IPI construction, averaging 58 per question on LongMemEval and 29 on LoCoMo; answer generation needs one additional call per query. In tokens, EvoMem uses 68.4K per question on LongMemEval and 36.8K on LoCoMo, against 45.4K and 97.6K for LightMem. IPI construction is a one-time cost per dialogue, whereas retrieval and answer generation are the only recurring costs, which keeps versioned evidence tracking affordable at deployment scale. Appendix C.5 analyses this cost–accuracy trade-off.

Offline EDPL calibration. Figure 3 traces accuracy and token-level F1 over three EDPL feedback iterations on the development split. Accuracy rises from 67.6% to 82.8% on LoCoMo and from 72.4% to 87.4% on LongMemEval, while token-level F1 rises from 30.2 to 42.0 and 38.2 to 43.6. Most of the improvement comes in the first iteration, raising accuracy by 11.4 points on LoCoMo and 10.4 on LongMemEval, so the calibration corrects a systematic misalignment. The test-set ablation confirms that the resulting frozen policy remains necessary: removing EDPL lowers test accuracy to 66.4% and 68.2%.

Error analysis. The remaining errors fall into three primary categories. First, retrieval misses occur when the necessary evidence exists in memory but is not retrieved, accounting for 54% of errors on LoCoMo and 60% on LongMemEval. Second, failures in resolving temporal expressions account for 31% and 20% of errors, respectively. Third, hallucinations involving facts absent from the available memory constitute the remaining 15% of LoCoMo errors and 20% of LongMemEval errors. Retrieval misses are the dominant failure mode on both benchmarks, suggesting that residual performance is primarily constrained by retrieval recall rather than by downstream re-ranking quality.

5 Conclusion

We introduced EvoMem, a memory architecture that separates versioned storage from query-adaptive retrieval, so that LLM agents can track changes in user facts and preferences across sessions. Type-adaptive retrieval selects evidence, scoring, and pruning strategies according to each query's requirements, while self-evolving consolidation preserves timestamped version chains and calibrates retrieval offline from past errors. Across LoCoMo and LongMemEval, EvoMem improves accuracy and F1 over the evaluated baselines under both routing settings. Future work will improve retrieval coverage and confidence estimation for long-horizon memory.

ETHICS STATEMENT

EvoMem creates an external memory of user interactions to track changing user attributes. The experiments use anonymized public datasets. IPI is restricted to explicitly stated, task-relevant, and revocable preferences, goals, decisions, and interaction constraints; it does not infer sensitive personal attributes. For future deployments, users will be informed about what data are stored and the retention period, and they will have a mechanism to inspect and delete their stored memories. The system should also apply data minimization, access controls, and safeguards against retaining incorrect information.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of this work, the detailed implementations of EvoMem are provided in Section 3, Appendix A, and Appendix B, including the hyperparameter configuration and the evaluation protocol. These measures are intended to facilitate the verification and replication of our results by other researchers in the field. The source code is available at https://anonymous.4open.science/r/Evomem/.

AI USE DISCLOSURE

In this work, we used generative AI tools for the experiment plans, part of the code implementation, and the polishing and grammar correction of the manuscript. We have not used generative AI tools to generate synthetic datasets or to formulate mathematical proofs. All AI-assisted work, including code, experimental results, and text, was reviewed and verified by the authors. We take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Nick Alonso, Tomas Figliolia, Anthony Ndirango, and Beren Millidge. Toward conversational agents with context and time sensitive long-term memory. *arXiv preprint arXiv:2406.00057*, 2024.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*, volume 2024, pp. 9112–9141, 2024.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *NeurIPS*, 38:113506–113543, 2026.

Yuxuan Cai, Yipeng Hao, Jie Zhou, Hang Yan, Zhikai Lei, Rui Zhen, Zhenhua Han, Yutao Yang, Junsong Li, Qianjun Pan, et al. Building self-evolving agents via experience-driven lifelong learning: A framework and benchmark. *arXiv preprint arXiv:2508.19005*, 2025.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *ACL*, pp. 16803–16822, 2026.

Ding Chen, Simin Niu, Kehang Li, et al. Halumem: Evaluating hallucinations in memory systems of agents. *arXiv preprint arXiv:2511.03506*, 2025.

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.

Renke Dai, Hebin Hu, Jiahui Zhang, Yilin Kang, and Ah-Hwee Tan. MemoryART: Enhancing LLMs via multi-memory models with adaptive resonance theory for healthcare agents. In *AAAI*, volume 40, pp. 20676–20683, 2026a.

Zijie Dai, Shiyuan Deng, Sheng Guan, Yizhou Tian, Xin Yao, Xiao Yan, and James Cheng. Recmem: Recurrence-based memory consolidation for efficient and effective long-running LLM agents. In *ACL Findings*, pp. 32353–32376. Association for Computational Linguistics, 2026b.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, ..., Zizheng Pan, and Zongqing Yao. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026.

Yiming Du, Bingbing Wang, Yang He, Bin Liang, Baojun Wang, Zhongyang Li, Lin Gui, Jeff Z Pan, Ruifeng Xu, and Kam-Fai Wong. Memguide: intent-driven memory selection for goal-oriented multi-session llm agents. In *AAAI*, volume 40, pp. 30584–30592, 2026.

Darren Edge, Ha Trinh, Andy Cheng, Joshua Jiang, Martha Larson, and Benjamin Marlin. 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, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation. In *ICLR*, 2026.

Daocheng Fu, Jianbiao Mei, Rong Wu, and Xuemeng Yang. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks. *arXiv preprint arXiv:2510.08002*, 2026.

Yunfan Gao, Yun Xiong, Xinyu Gao, et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, et al. Memory in the age of ai agents: A survey. *arXiv preprint arXiv:2512.13564*, 2026.

Zhaopei Huang, Qifeng Dai, Guozheng Wu, Xiaopeng Wu, Xubin Li, Tiezheng Ge, Wenxuan Wang, and Qin Jin. Mem-PAL: Towards memory-based personalized dialogue assistants for long-term user-agent interaction. In *AAAI*, volume 40, pp. 31229–31237, 2026.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. Magma: A multi-graph based agentic memory architecture for ai agents. *arXiv preprint arXiv:2601.03236*, 2026.

Sangyeop Kim, Yohan Lee, Sanghwa Kim, Hyunjong Kim, and Sungzoon Cho. Pre-storage reasoning for episodic memory: Shifting inference burden to memory for personalized dialogue. *arXiv preprint arXiv:2509.10852*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*, volume 33, pp. 9459–9474, 2020.

Dongfang Li, Zetian Sun, Xinshuo Hu, Baotian Hu, and Min Zhang. CMT: A memory compression method for continual knowledge learning of large language models. In *AAAI*, volume 39, pp. 24413–24421, 2025.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *ICLR*, volume 2024, pp. 52989–53046, 2024.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *ACL*, pp. 13851–13870, 2024.

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

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *UIST*, pp. 1–22, 2023.

Chen Qian, Xin Cong, Cheng Yang, Wei Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. Chatdev: Communicative agents for software development. In *ACL*, 2024.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *EMNLP*, pp. 3982–3992, November 2019.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *ICLR*, volume 2024, pp. 32628–32649, 2024.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nicolas Sch"arli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *ICML*, volume 202. PMLR, 2023.

Noah Shinn, Federico Cassano, Anjali Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *NeurIPS*, 36, 2023.

Cassandra Hui-Ming Tan, Budhitama Subagdja, and Ah-Hwee Tan. ARTEM: Enhancing large language model agents with spatial-temporal episodic memory. In *AAAI*, volume 40, pp. 25753–25760, 2026a.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, Liming Zhu, and Wenjie Zhang. Memotime: Memory-augmented temporal knowledge graph enhanced large language model reasoning. In *WWW*, pp. 4220–4231, 2026b.

Zihao Tang, Xin Yu, Ziyu Xiao, Zengxuan Wen, Zelin Li, Jiaxi Zhou, Hualei Wang, Haohua Wang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. Mnemis: Dual-route retrieval on hierarchical graphs for long-term llm memory. In *ACL*, 2026.

Guanzhi Wang, Yunfan Xie, Yifan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, and Wang-Cheng Kang. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *ICLR*, 2025.

Zhaofen Wu, Hanrong Zhang, Fulin Lin, Wujiang Xu, Xinran Xu, Yankai Chen, Henry Peng Zou, Shaowen Chen, Weizhi Zhang, Xue Liu, Philip S. Yu, and Hongwei Wang. GAM: Hierarchical graph-based agentic memory for LLM agents. In *ICLR*, 2026.

Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Wenlin Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, and Tong Xu. From single to multi-granularity: Toward long-term memory association and selection of conversational agents. In *ICLR*, 2026a.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *NeurIPS*, 38:17577–17604, 2026b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In *AAAI*, volume 38, pp. 19632–19642, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*, volume 36, 2024.

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

A Method Reproduction Details

A.1 Query Profiling and Component Activation

The profile vector $\mathbf{p}(T_q) = (f_s, t_s, d_i)$ is read from the type T_q. Each type is mapped to a fixed profile, so no additional inference-time computation is required after the type decision. FEI and FG are the default structures. Fact-sensitive types additionally consult EAT and IPI, while temporally sensitive types follow the relevant FG version chain. EDPL can modify HBS ranking when a learned pattern applies to the query. In the predicted setting, T_q is produced by a lightweight router (Appendix B.6). In the gold setting, T_q is the benchmark-annotated type.

Table 5: Query-type to retrieval-profile mapping used by the fixed table of Eq. 2. The profile is (f_s, t_s, d_i); every query starts from FEI and FG, and the extra structures are added by the flags f_s (explicit facts), t_s (version-chain traversal), and d_i (hierarchical layer traversal).

Type	Profile	Extra structures	Retrieval path
SSU	(1,0,0)	EAT, IPI	FEI exact match plus explicit facts and persona entries
SSA	(1,0,0)	EAT, IPI	FEI exact match plus explicit facts and persona entries
SSP	(1,0,0)	EAT, IPI	FEI plus explicit facts and persona preference entries
TR	(1,1,0)	version chain	FG version chain ordered by date
KU	(1,1,0)	version chain	FG version chain, newest supported version first
MS	(1,0,1)	layer traversal	FEI plus hierarchical layer traversal for cross-session aggregation

A.2 Heuristic Beam Search Scoring

For every development query q of type T_q, let $C(q)$ be its FG candidate nodes and let $R^*(q) \subseteq C(q)$ be the minimum provenance-linked support set from which the reference answer can be derived. We train the HBS parameters $\theta_{T_q} = (\alpha_{T_q}, \beta_{T_q}, \gamma_{T_q}, \delta_{T_q}, w_{T_q})$ with the support-evidence ranking loss of Eq. 4, in which the regularizer prevents a sparsely represented query type from drifting far from the shared policy. Thus, the objective increases the total HBS score of evidence that supports the reference answer relative to all candidates retrieved for the same question. The development split supplies only ranking supervision, and the learned parameters are frozen for static test evaluation. EDPL may subsequently adjust the corresponding weights after a failure identified through feedback, as described in Appendix A.4.

A.3 Hyperbolic Decay Pruning

The temporal-decay rate is type-dependent and set to 0.003 for knowledge-update and preference questions, 0.008 for temporal-reasoning and multi-session questions, and 0.02 for single-session questions; the access-recency coefficient is fixed at $\lambda = 0.01$. The pruning threshold $\theta_{T_q}^{\text{prune}}$ is 0.2 for knowledge-update, preference, and multi-session questions and 0.6 for temporal-reasoning and single-session questions. The decay term is evaluated against the time that the query targets rather than the time the query is issued whenever the question specifies a target time, so historical questions are not penalized for asking about an older value, and the query time is used only for questions without a target time. Pruning is applied to the retrieved candidate set at query time rather than to storage, so the version chains remain complete and the filter is query-adaptive.

A.4 Error-Driven Pattern Learning

EDPL operates in three steps after a failure identified through feedback. (1) *Failure collection.* The completed query, system response, feedback signal, and retrieved set R (Eq. 5) are recorded. (2) *Counterfactual diagnosis.* The minimum evidence set R^* is the smallest provenance-linked support set of FG nodes from which the corrected answer a_q^* can be derived. The diagnosis identifies the retrieval dimension $d \in \{\text{structural}, \text{semantic}, \text{confidence}, \text{temporal}\}$ in which R and R^* differ most. A rule template converts this difference into an instruction ι_q. In deployment, the reference evidence set $R^*(q)$ is not directly observable, so the diagnosis approximates it from the correction implied by the user's feedback, localizing the node whose value contradicts the corrected answer and falling back to the retrieved set when the correction does not identify a specific fact. (3) *Pattern update.* Each pattern $p = (\mathbf{t}_q, \iota_q, c_q)$ stores trigger keywords $\mathbf{t}_q$ defined at the level of query types and retrieval dimensions, an instruction ι_q, and a Beta-Binomial confidence $c_q = \frac{1+s}{2+s+f}$ over successes s and failures f. Each later query influenced by an active pattern updates s or f when feedback becomes available. Patterns below the confidence threshold are disabled. Duplicate failures with the same trigger keywords and diagnosis type merge into one pattern. The diagnosis also adjusts the relevant HBS weights for later retrieval. Development examples provide feedback in the static benchmark protocol, and learning is switched off before test evaluation. In deployment, the same process runs only after feedback on a completed interaction.

A.5 Complexity Derivations

For storage, the FEI stores N vectors of dimension d, requiring $O(Nd)$ space. The FG stores $|V_{\text{attr}}|$ attribute nodes and a chain of version nodes for each attribute. Each observation appends one node, so $\sum_v |\text{chain}_v| = O(M)$ where M is the number of observations. The EAT fact store uses $O(|\mathcal{F}|)$ space. For retrieval, FEI top-k selection costs $O(Nd)$ with exact cosine search or $O(\log N \cdot d)$ with an ANN index. HBS expands a beam of width b over chains of length L, costing $O(bLd)$ for each matched entity. Pruning evaluates a constant number of score terms per candidate. The worst-case per-query retrieval cost is $O(Nd + |\mathcal{C}_{\text{active}}| \cdot bLd)$, independent of total history length except through the number of sessions N. LLM cost is one call per query for answer generation, one IPI call per session, and one EDPL diagnosis call for each completed interaction that receives an error signal and feedback. In static benchmark evaluation, EDPL calls are restricted to the development split.

A.6 Fact Graph Construction

The FG is built offline from the facts extracted by EAT, the session texts, and the session dates, and it issues no LLM calls. Construction proceeds in three passes.

Attribute and version nodes. Each fact $f = (s, p, o_v, t, c, \text{src})$ is inserted under its attribute key (s, p). If a version node with the same value and date already exists under that key, its count of supporting source sessions increases by 0.5. Otherwise, a new version node is appended to the attribute. A version node stores the value, date, extraction confidence, number of supporting source sessions, and source session. Attribute nodes identify the queryable property, and version nodes record the values that property has taken over time.

Edges. Ownership edges connect an entity to each of its attribute keys, chronological edges connect consecutive version nodes of the same attribute after sorting by date, and provenance edges connect each version node to the session it came from. The chronological edges are what make an earlier value addressable after a later one has been recorded; no value is overwritten.

Graph layers. FG uses four functional layers. L3 contains semantic-topic nodes that organize related attributes. L2 contains entities and their attribute hubs. L1 contains chronologically ordered version nodes for each entity-attribute pair. L0 contains the session and turn records that provide provenance for every version. Entity nodes are seeded by subjects in extracted facts and extended with speakers from turn tags and capitalized noun phrases, after a stop list removes pronouns and question words. Semantic-topic nodes are linked to the attribute hubs they organize; entity nodes and attribute hubs are connected by ownership edges; chronological edges link consecutive L1 versions; and provenance edges link each version to its supporting L0 record.

At retrieval time a query profiled as inference-heavy traverses the layers top-down. Year matches add 3, month matches 5, season matches 4, event-type matches 4, and entity matches 3 to the score of each candidate session, and the 20 highest-scoring sessions are added to the context. Layer traversal

is a second route to the same episodes that the FEI reaches by embedding similarity, and the two routes are merged before answer generation.

A.7 Session-Level Intent and Preference Profiles

IPI issues one LLM call per session. The input is the full session text with turns marked as `[user]` and `[assistant]`, together with the session date, and the model returns one line per field in the form `field: value`. It extracts only information explicitly stated by the user that is relevant to a task or interaction. Fields with no direct evidence are returned as `N/A` and are not stored. Table 6 lists the four permitted fields and the evidence each one may use.

Table 6: The four explicit and task-relevant fields extracted by IPI.

Field	Signals the model is asked to use
preferences	explicitly stated likes, dislikes, and choices relevant to the interaction
goals	explicitly stated objectives, plans, or desired outcomes
task decisions	decisions, commitments, and selected options stated in the dialogue
interaction constraints	explicit requirements, restrictions, formats, or timing constraints

The prompt π_{persona} is given below.

```
Extract only explicit, task-relevant, and revocable information
stated by the user. Do not infer personality, demographic,
financial, health, political, religious, relationship, or location
attributes.

Session date: <session date>

Conversation:
<session text; turns marked [user] and [assistant]>

For each of the following 4 fields, copy or concisely normalize
only information explicitly stated by the user. If no direct
evidence exists, write "N/A".

Dimensions:
  - preferences: Explicit likes, dislikes, choices, or stated
    priorities relevant to the interaction.
  - goals: Explicit objectives, plans, or desired outcomes.
  - task_decisions: Explicit decisions, commitments, and selected
    options.
  - interaction_constraints: Explicit requirements, restrictions,
    response formats, or timing constraints.

Output format (one per line, NO extra text):
preferences: <value>
goals: <value>
task_decisions: <value>
interaction_constraints: <value>
```

How session-level profiles support cross-session queries. Profiles are stored per session and are never merged into a single user-level summary, because the memory holds one profile per session, $\mathcal{P} = \bigcup_i \mathcal{P}(S_i)$. A preference-, goal-, or task-oriented query retrieves the three profiles whose sessions are most similar to it, above a cosine threshold of 0.2, and adds them to the context with their session dates. Each entry remains attributable to its session and date, and users can revoke entries when they no longer want them retained.

In addition, every non-empty field is written into FG as an attribute with predicate `ipi_<field>`, timestamped with the session date, so an explicitly stated preference, goal, decision, or constraint can form a version chain alongside rule-extracted attributes. These entries are used as descriptive evidence rather than as update signals; they are excluded from the value-change detection that decides whether an attribute was updated, which would otherwise read a repeated description as a new value.

The construction cost is one LLM call per session, and the number of persona-inferred sessions is bounded by a configuration cap, which trades construction cost against accuracy (Appendix C.5).

B Experiment Protocol

B.1 Datasets and Evaluation Subsets

LoCoMo Maharana et al. (2024) contains long-term multi-session dialogues spanning up to 35 sessions, grounded in temporally structured event graphs; its questions are labeled with four categories (multi-hop, temporal, open-domain, and single-hop), and following common practice we exclude the adversarial category. LongMemEval Wu et al. (2025) provides curated questions across six types (SSU, SSA, SSP, TR, KU, MS) over scalable user-assistant interaction histories. Accuracy results are reported on the evaluated subset of each benchmark (Section 4.1), which is disjoint from the development splits used for tuning (Appendix B.6); no test question contributes to any tuning decision. Per-question cost figures (Table 7) are averaged over a separate sample of 100 questions per benchmark, so that token and call counts are comparable across methods. The LongMemEval slice with changing attribute values used for Table 4 contains 17 attributes across 8 conversations, split into three categories, namely questions about current values (the most recent value), historical-value questions (an earlier value), and conflict questions (two or more conflicting observations). The FullText-30K baseline receives the first 30,000 characters of each conversation, the same window on which EvoMem's retrieval operates.

B.2 Baselines and Reimplementation

All baselines are reimplemented with the DeepSeek-V4-Flash backbone and evaluated under an identical protocol (same judge, same evaluation code). BM25 RAG indexes episode chunks with BM25 and retrieves the top-k chunks per query. MemoryBank Zhong et al. (2024) summarizes conversation history periodically with an LLM and applies a forgetting curve for retrieval prioritization. PREMem Kim et al. (2025) pre-extracts factual, experiential, and subjective fragments during memory construction and retrieves them at inference. Mnemis Tang et al. (2026) combines System-1 embedding-based retrieval with System-2 hierarchical graph traversal over a four-layer concept hierarchy. RecMem Dai et al. (2026b) consolidates recurring information into summary memory through semantic refinement. LightMem Fang et al. (2026) pre-compresses input tokens, organizes content into topic-aware short-term memory, and performs offline sleep-time consolidation. Where a baseline's official code is unavailable, we follow the description in its paper and tune its hyperparameters on the development split; all baselines were given equivalent development data for tuning.

B.3 Implementation Configuration

Generation uses DeepSeek-V4-Flash DeepSeek-AI et al. (2026) via the DeepSeek API (base URL: https://api.deepseek.com/v1; model identifier: deepseek-v4-flash) during 9 to 11 August 2026 (UTC+8), temperature 0.0, fixed seed 42. Judging uses GLM-5.3-Flash as an independent judge model. All embeddings use SentenceTransformers with BAAI/bge-small-en-v1.5 (384 dimensions). FEI top-k: 80 for temporal types, 100 for multi-session, 60 for other types. HBS beam width: type-dependent 1 to 5 (KU/MS: 5, TR/SSP: 3, SSU/SSA: 1); confidence-calibration pseudo-count $\kappa = 3$; pruning thresholds $\theta_{T_q}^{\text{prune}} \in \{0.2, 0.6\}$ (KU/SSP/MS: 0.2, TR/SSU/SSA: 0.6); type-dependent temporal decay $\eta \in \{0.003, 0.008, 0.02\}$ (KU/SSP: 0.003, TR/MS: 0.008, SSU/SSA: 0.02); access-recency $\lambda = 0.01$. EAT rule library: five categories (action, attribute, count, person, frequency) with syntactic templates; no inference-time LLM calls. IPI: one LLM call per session and four explicit, task-relevant, and revocable fields. EDPL: three development iterations, pattern confidence via Beta-Binomial update, triggers restricted to query types and retrieval dimensions.

B.4 Evaluation Metrics

LLM-as-a-Judge accuracy is computed by a single fixed GLM-5.3-Flash judge that receives the question, the ground-truth answer, and the system response, and returns a binary correctness verdict; the judge prompt is identical across all methods and settings. Before the judge call, a deterministic pre-check rejects empty answers, explicit no-information replies, template artifacts, and polar one-

word answers to long-form questions, and accepts exact numeric matches, relative-date matches within one day, and high keyword overlap, so only the remaining items are sent to the judge. The judge template is reproduced in Appendix B.5. Token-level F1 is computed over tokenized generated and ground-truth answers, counting common tokens with the standard F1 formula; per-category values average the questions of each category, and overall values are reported over each benchmark's test questions. All numbers in the tables were produced by the same aggregation script (provided in the reproducibility package).

B.5 JUDGE PROMPT

The judge receives the question, the expected answer, and the predicted answer, and replies with a single verdict. The template is reproduced verbatim below; only the three placeholders are filled per item.

```
Evaluate whether the predicted answer conveys the SAME ESSENTIAL INFORMATION
as the expected answer.
Question: {question}
Expected: {gold}
Predicted: {predicted}

Accept YES if:
- Same meaning with different wording
  (e.g., "not considered religious" == "somewhat not extremely")
- Date accepted: "2023-06-27" is the same as "27 June 2023"
  or "the week of 27 June 2023"
- Slight numeric difference: "1" accepted if the expected value is "2"
  and the context mentions limited events
- Partial but correct entity name or activity
Reply ONLY YES or NO.
```

B.6 ROUTER AND DEVELOPMENT PROTOCOL

In the predicted setting, query types are produced by a lightweight router. On LongMemEval the router is a zero-shot LLM that applies a fixed decision order over surface cues, which cover recommendation or advice requests, reminders of an earlier assistant reply, temporal arithmetic, current-value or frequency wording, and cross-session counting or summation, together with few-shot examples drawn from a held-out pool that excludes the evaluated questions. It attains 91.6% classification accuracy over the six LongMemEval types. On LoCoMo the query is embedded with the same bge-small-en-v1.5 encoder, and a logistic-regression head (trained on the development split) maps the embedding to one of the benchmark's query types. In both cases the router output is taken as the predicted type. The development split is disjoint from the test questions; for LoCoMo the split is made at the dialogue level so that all questions of a held-out dialogue stay in the same split; the same split is used for EDPL pattern learning, HBS weight tuning, and pruning-threshold calibration. All systems, including baselines, received equivalent development data for any tuning they required. The gold setting uses the benchmark-annotated types directly; Table 3 bounds the value of the annotations by comparing gold with predicted, uniform, soft, random, and shuffled routing.

The routing strategies compared in Table 3 differ in how the query type is obtained or applied. *Uniform* routing ignores the query and applies one fixed profile to every query, so all evidence structures and ranking controls are activated with the same weights. *Soft* routing keeps the classifier but replaces the argmax decision with a probability-weighted combination of the type-specific profiles, so each query is served by a mixture of profiles rather than a single one. *Random* routing draws a type uniformly at random from the type set of the benchmark for each query, independently of the query. *Shuffled* routing keeps the predicted labels but permutes them across the questions of the benchmark, so each query receives the type of another query, which preserves the label distribution while removing the correspondence between query and type.

C Additional Diagnostics

C.1 Per-Category Analysis

All per-category values in this section are the ones reported in Table 1. On LongMemEval, EvoMem (gold) reaches 88.0/47.6 on SSU, 100.0/50.9 on SSA, 82.2/43.4 on SSP, 91.2/43.0 on TR, 87.0/38.6 on KU, and 82.0/43.5 on MS. Mnemis is the strongest baseline at 83.2/38.2 overall, benefiting from dual-route coverage, but it applies the same pipeline to all question types and overwrites prior attribute values Tang et al. (2026), which destroys the temporal context needed for KU and TR queries. The largest per-category accuracy gaps over Mnemis are on KU, 87.0 against 70.0, and TR, 91.2 against 80.0. BM25 RAG peaks on SSA at 90.0/32.3 through lexical matching but drops to 50.0/23.1 on KU and 55.0/23.1 on TR. MemoryBank's consolidation loses granularity for precise factual recall, and PREMem propagates extraction errors downstream. On LoCoMo, EvoMem (gold) reaches 86.6/57.0 on single-hop, 83.4/43.4 on temporal, 80.0/33.6 on multi-hop, and 80.0/36.0 on open-domain questions, which is between 5.0 and 8.6 accuracy points above the strongest baseline in every category; the single-hop margin of 86.6 against Mnemis at 79.0, together with the temporal gain to 83.4 from 77.0, drives the overall improvement of 6.2 points. Open-domain questions show the narrowest F1 margin, since the answers include the supporting evidence that token-level F1 discounts, indicating that separating the committed answer from its justification is the remaining measurement gap on this benchmark.

C.2 Evidence-Scope Analysis

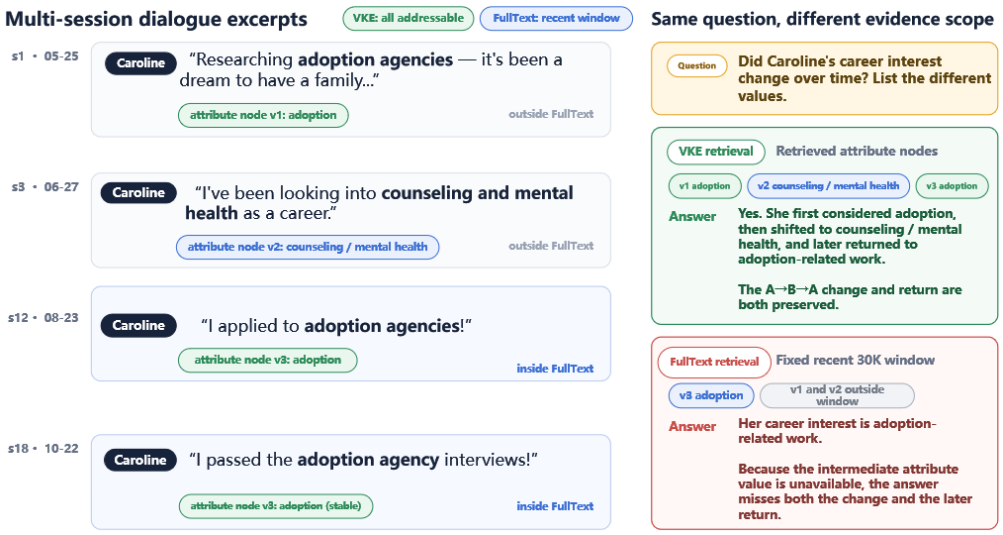

Figure 4: Retrieval paths of the FullText-30K baseline and EvoMem over an attribute's version chain.

Figure 4 illustrates why the FullText-30K baseline fails on attributes whose values change over time, and Table 4 reports the per-category numbers. Specifically, the raw conversational window contains every mention of the attribute, including outdated values; without a version chain, the model cannot tell which mention is current. EvoMem's retrieval path instead follows the version chain and returns the relevant node for the question type (current, historical, or conflict). The largest margin over FullText therefore occurs on questions about current values (+29.4) and the smallest on conflict questions (+12.4), where the raw text itself is ambiguous but partially informative.

C.3 EDPL Learning Curves

Figure 3 (Section 4.3) reports three EDPL feedback iterations on the development split. On LoCoMo, accuracy rises from 67.6% to 82.8% and token-level F1 from 30.2 to 42.0; on LongMemEval, the corresponding values rise from 72.4% to 87.4% and from 38.2 to 43.6. These curves describe the offline calibration process, not an independent test-set result, because the same development split is used to learn the patterns. Table 2 reports the complementary test-set ablation of EDPL.

C.4 Error Analysis

Manual inspection of the remaining errors on both benchmarks reveals three recurring patterns. Retrieval misses are the most frequent, accounting for 54% of errors on LoCoMo and 60% on LongMemEval. The required evidence exists in the conversation history but is not retrieved, and the model either abstains with "Not mentioned" or returns partial coverage. For instance, the question "Which meat does Audrey prefer eating more than others?" (answer: "chicken") yields "Not mentioned," although the fact is explicitly stated in the conversation. Temporal expression failures constitute the second pattern (31% on LoCoMo, 20% on LongMemEval), mostly relative-time vagueness, where an absolute-date question is answered with a relative expression (e.g., "Last week" for "June 2023"), or a date stated in the history is not recovered. Hallucination is the third pattern (15% on LoCoMo, 20% on LongMemEval), in which the model fabricates facts absent from the conversation (e.g., "Metal detecting" for a hobby whose ground truth is "Extreme sports"). These patterns stem from two underlying limitations. First, implicitly expressed answers leave queries with little lexical or semantic overlap with the supporting evidence, so the relevant facts are never retrieved into the candidate set. Error-driven learning re-ranks the candidates that are retrieved but cannot recover evidence missed at the recall stage. Second, temporal anchoring is brittle for relative expressions, since mapping references such as "last week" to absolute dates must survive both retrieval and generation. These findings motivate the future directions discussed in Section 5: retrieval-completeness estimation, session-level date resolution, and segment-level confidence marking.

C.5 Cost and Latency Details

Table 7: Per-question API usage on the 100-question subsets of LongMemEval and LoCoMo (DeepSeek-V4-Flash).

	LongMemEval		LoCoMo	
	Calls	Tokens	Calls	Tokens
EvoMem	58	68.4K	29	36.8K
LightMem	23	45.4K	33	97.6K
FullText	1	104.9K	1	20.8K

We compute per-question API usage by dividing total offline memory-construction cost by the number of evaluated questions; it does not imply that the system rebuilds memory for every query. EvoMem's cost is dominated by offline IPI construction, which reads each session once to produce a persona profile. Retrieval and answer generation require one LLM call per query.

The measurements count all LLM calls per method over the 100-question subsets used in Table 7. For EvoMem, the dominant term is offline ingestion, where IPI contributes one call per persona-inferred session (roughly 58 per question on LongMemEval), whereas retrieval and answer generation together cost one call per query. On LoCoMo the per-question profile is lower (29 calls per question) because fewer sessions are persona-inferred. LightMem uses fewer tokens than EvoMem on LongMemEval (45.4K per question, 23 calls) but more on LoCoMo (97.6K per question, 33 calls), reflecting its summarization-heavy consolidation on longer dialogues. FullText answers each question from the full conversation in a single call, so its token count reflects prompt length rather than the number of calls (104.9K tokens per question on LongMemEval). Because IPI is an offline cost, it can be traded off against accuracy. Restricting persona inference to a bounded number of sessions reduces construction cost while retaining most of the gain (removing IPI drops accuracy to 72.0 on LoCoMo and 73.8 on LongMemEval, Table 2). The complexity bounds in Section 3.4 and Appendix A.5 hold under this trade-off.

The asymmetry between the two benchmarks follows from the two systems' opposite cost structures. EvoMem's token consumption is dominated by offline, session-level ingestion (IPI makes one call per session and reads the full session text), so its per-question cost tracks how many sessions the evaluated questions touch, at 58 calls per question on LongMemEval versus 29 on LoCoMo. LightMem's cost instead tracks a turn-level pipeline, where every turn is pre-compressed, and the short-term buffer triggers a summarization call (with both input and output tokens) whenever it fills. LoCoMo's dialogues are the longest in either benchmark, with up to 35 sessions of dense turns, so summarization is triggered more often and each call processes more tokens, pushing LightMem's

total above EvoMem's one-pass-per-session ingestion; on LongMemEval, fewer summarization triggers keep LightMem's total below EvoMem's.

D Limitations and Boundary Conditions

We state the boundary conditions of the empirical claims in this paper.

Query-type annotations. The upper-bound setting, in which EvoMem (gold) routes queries using the benchmark-annotated types, measures the value of those annotations. These annotations encode what evidence a query requires, not answer content, and the routing strategy comparison in Table 3 shows that the gain requires correctly recognized query types (gold $>$ predicted $>$ uniform $>$ random $\geq$ shuffled). The predicted setting demonstrates that a lightweight classifier retains most of the gain, at 80.0 on LoCoMo and 85.2 on LongMemEval against 82.6 and 88.4 under gold routing; the residual gap bounds the headroom for a stronger learned router. We did not report a full confusion-matrix analysis of the router, and its per-class behavior may vary by category.

Feedback-driven pattern learning. For static benchmark evaluation, patterns and HBS weights are learned from answer-supervised feedback on a held-out development split, then frozen before testing. This makes test-time behavior independent of test labels. In deployment, EDPL can continue to learn after user feedback on a completed interaction, updating patterns and HBS weights only for subsequent queries. The self-evolving claim therefore refers to feedback-driven policy updates over time, not access to the answer of the current query. All baselines received equivalent development data for tuning in the static protocol.

Statistical scope. Results are reported for a single backbone (DeepSeek-V4-Flash), a single seed (42), and a single judge model (GLM-5.3-Flash), without confidence intervals or significance tests. Temperature 0.0 reduces but does not eliminate sampling variance in API-based decoding and judging. Per-category values in Table 1 average the questions of each category over the test portion of each benchmark. The evidence-scope comparison (Table 4) covers 17 attributes, so a single attribute contributes about 5.9% of the overall score, and we treat that comparison as qualitative.

Coverage weakness. On LoCoMo, accuracy leads the strongest baselines in every category, yet the F1 margin narrows on open-domain questions, where EvoMem (gold) reaches 36.0 F1 against 36.5 for LightMem, since the answers include the supporting evidence that token-level F1 discounts; separating the committed answer from its justification remains the main residual gap on this benchmark, and we expect retrieval-completeness estimation to address it.

Scope of the FullText control. The FullText-30K baseline receives the first 30,000 characters of each conversation, a bounded window rather than the full dialogue. The comparison isolates the effect of versioned tracking and adaptive retrieval over an identical window; it does not claim that full-context models would perform at the same level. Because the window is taken from the start of each conversation, attributes whose values change later in the dialogue are truncated for the control, and a recency-window variant is left for future work.