GRAPHCARD: A SIMPLE GRAPH MEMORY PLUG-IN FOR LONG-TERM CONVERSATIONAL MEMORY SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Conversational memory in deployed systems is predominantly vector-based: memories are stored as extracted facts or raw dialogue chunks, and a query is answered through fast per-entry similarity search. Production favors this design for its speed, scalability, and strong overall accuracy. However, multi-hop and temporal questions depend on relations and timelines that live *between* entries, and per-entry retrieval cannot see them. A *graph* memory captures exactly this missing layer, linking entities through facts. The two designs are natural complements: vector retrieval buys coverage, graph structure buys relations. Their combination, however, remains underexplored: existing graph designs *replace* the vector stack rather than augment it, and we find they score far below the strongest vector-based systems they aim to improve. In this paper, we take the strengths of both by augmenting rather than replacing: the existing vector memory (the *host*) stays untouched, and GRAPHCARD, a graph plug-in, runs beside it. At memory ingestion time, a plug-in branch runs parallel to the host's own pipeline, organizing the raw conversation stream into a fact graph through entity deduplication and temporal fact management. At search time, a single query embedding and a bounded graph search, with no LLM call, quickly condense the relevant subgraph into one $\sim$150-token *graph card* prepended on top of the host's retrieved memories. Nothing else in the host memory system changes, yet the missing relational layer is now in the context. Extensive experiments across two memory benchmarks (LoCoMo, LongMemEval), two model groups (GPT, Claude), and four strong vector-based baseline hosts (RAG, Nemori, SimpleMem, Mem0) show that GRAPHCARD improves every host configuration, by up to +12.3 points, with gains concentrated on multi-hop and temporal questions. GRAPHCARD also yields a better accuracy–token trade-off than widening the host's own retrieval: against Mem0, GRAPHCARD is $\sim$1.5 points more accurate at a matched memory budget; it matches the same host's accuracy with 79% fewer additional tokens. GRAPHCARD is fast as well, with an LLM-free search that completes in 0.61 s at p50, in the same sub-second range as the vector hosts themselves.

1 INTRODUCTION

Conversational assistants increasingly rely on long-term memory: information mentioned in one session must be retrievable many sessions later (Xu et al., 2022a;b; Park et al., 2023; Zhang et al., 2025). The designs that dominate production are *vector-based*: extractive memory systems such as Mem0 (Chhikara et al., 2025) condense dialogue into independent fact entries, while retrieval-augmented approaches (Lewis et al., 2020) index raw dialogue chunks. The choice is well motivated: per-entry embedding retrieval is fast enough for interactive use, scales with conversation length, and answers most questions well (Chhikara et al., 2025; Liu et al., 2026; Patel & Patel, 2025). However, it cannot see *between* entries. Each candidate is scored independently against the query, yet real memories are related: people, events, and dates form a graph. A question like "did she change jobs after moving?" is answered by no single entry. The answer lies in the connection between two facts and in their order in time. Such a retriever can fetch each fact but never the connection between them, and ranking by similarity alone rarely retrieves the two facts together.

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.

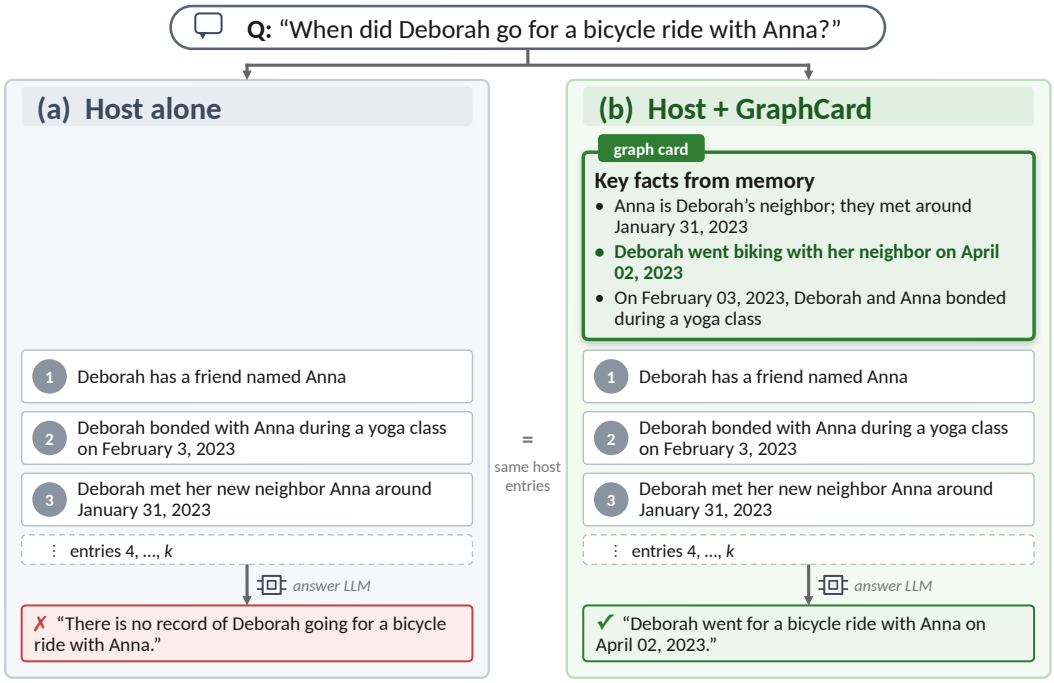


Figure 1: Illustration of graph memory plug-in behavior. The host's retrieval pipeline and its entries are identical on both sides. The only intervention is the graph card added on top of the host's list, which flips the host's answer from "no record" to the exact date.

The structure those questions need is exactly what *graph memory* (Yang et al., 2026) makes explicit: conversational facts organized as a knowledge graph, related entities linked, time attached. The two designs are natural complements, vector buying coverage, graph buying relations, and the question is how to combine their strengths. That combination remains underexplored. Existing graph systems are *replacement* designs: the graph becomes the primary retrieval path, and the answerer sees graph-extracted facts *instead of* the dialogue evidence itself. Replacement carries two costs: (1) Coverage: a fact reduced to a triple loses the wording, the qualifiers, and the surrounding detail that answer the long tail of questions, and the graph stores only what the extractor keeps. (2) Speed: the traditional query path for a graph includes entity extraction, which requires LLM calls and adds latency to every question. Our controlled re-evaluation of three representative graph memory systems, Mem0^g (Mem0's official graph variant), Graphiti (Rasmussen et al., 2025), and A-MEM (Xu et al., 2026), finds that, with LLMs as of 2026 (*e.g.*, GPT-5.6 Luna (OpenAI, 2026), Claude Haiku 4.5 (Anthropic, 2025)), all three fall well short of the strongest vector-based systems they aim to improve.

In this paper, we address this problem by realizing the combination with a **graph memory plug-in**: graph structure is added *next to* the existing vector-based memory system (the *host*), rather than replacing it. Concretely, the graph branch should be (i) *input-decoupled*: it consumes the raw conversation stream directly, assuming nothing about the host's internal memory format, so the same plug-in attaches to any host without modification; and (ii) *output-attributable*: the host's retrieval pipeline stays untouched, and graph information enters as a single card prepended on top of the host's list, so in production, any change in the results can be attributed to exactly that card, which keeps retrieval analysis and evaluation simple. We implement this design as GRAPHCARD, a *simple* plug-in that delivers relational information to existing memory systems. At ingestion time, GRAPHCARD runs in parallel with the host's own pipeline, and turns the raw conversation into a fact graph through three construction steps. Specifically, an LLM first distills each incoming turn into dated facts that connect the entities they mention. To resolve entity duplication, an LLM merges the same person or place appearing under different names into one node, keeping the graph clean. To manage temporal information, relative expressions such as "yesterday" are rewritten as actual dates at storage time, and when a new fact supersedes an old one, the old fact is marked as outdated rather than deleted

(similar to Zep/Graphiti (Rasmussen et al., 2025)). At search time, through a single query embedding and a bounded graph search, GRAPHCARD retrieves the relevant subgraph, and organizes it into one ∼150-token *graph card*, which is prepended to the host's retrieved list (Figure 1).

We conduct extensive experiments across two benchmarks (LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2025a)), two model groups (GPT, Claude), and four strong memory hosts (RAG, Nemori, SimpleMem, Mem0). GRAPHCARD improves *every* host configuration, by up to +12.3 points, and the gains concentrate on multi-hop and temporal questions, exactly where per-entry retrieval falls short. GRAPHCARD also yields a better accuracy–token trade-off than widening the host's own retrieval: against Mem0 on LoCoMo, it matches the accuracy of top-30 retrieval with 79% fewer additional tokens. An interesting finding is that GRAPHCARD-alone scores below almost every host, yet prepending card information improves every one of them. This indicates that GRAPHCARD and the existing host are complementary rather than redundant: GRAPHCARD contributes structure the hosts lack, and the hosts contribute coverage GRAPHCARD lacks.

Our contributions are threefold:

- We propose GRAPHCARD, a host-agnostic graph memory plug-in, to bring relations into mature vector-based memory systems, strengthening their ability to answer multi-hop and temporal questions. GRAPHCARD attaches to the host memory as a plug-in branch: it reads only the raw conversation stream and adds just one ∼150-token graph card on top of the host's list.
- We design a set of extraction and query strategies for graph memory. At ingestion time, we deduplicate entities through LLM decisions and resolve relative time into absolute dates. At search time, we match the question directly against the graph with one query embedding and a bounded graph search, which involves no LLM call and completes in only 0.61 s at p50.
- We evaluate GRAPHCARD spanning two benchmarks, two model groups, and four memory hosts, and it improves every host configuration by up to +12.3 points. Notably, GRAPHCARD-alone scores below almost every host, yet adding it improves the accuracy of every host. This shows that the card helps not by being a replacement, but by supplying what vector retrieval misses. Tokens spent on the card also buy more accuracy than tokens spent on wider retrieval.

2 RELATED WORK

2.1 ENTRY-BASED CONVERSATIONAL MEMORY

Entry-based systems store and manage dialogue history as discrete entries. The simplest is chunk-based RAG (Lewis et al., 2020; Karpukhin et al., 2020), which indexes raw dialogue chunks and retrieves them by similarity; MemoryBank (Zhong et al., 2024) adds forgetting-aware updates; MemGPT (Packer et al., 2023) manages memory through agentic paging; SeCom (Pan et al., 2025) constructs memory at segment granularity, between the turn and session levels; Mem0 (Chhikara et al., 2025) extracts key facts and consolidates them into persistent items; Nemori (Ma et al., 2026) segments dialogue into episodes and derives semantic memories, resolving relative time at write time; SimpleMem (Liu et al., 2026) condenses interactions into compact multi-view indexed units. These systems differ in how entries are formed, but retrieval remains *per-entry*: each candidate is scored independently. All of these systems can serve as the hosts that GRAPHCARD plugs into.

2.2 GRAPH-CENTRIC CONVERSATIONAL MEMORY

Graph-centric systems make a graph the primary memory store. Zep/Graphiti (Rasmussen et al., 2025) maintains a bi-temporal knowledge graph with edge invalidation; A-MEM (Xu et al., 2026) links Zettelkasten-style notes and expands retrieval along those links; Mem0^g (Chhikara et al., 2025) stores entity–relation edges and retrieves over them; MAGMA (Jiang et al., 2026) retrieves by policy-guided traversal over multi-view (semantic, temporal, causal, entity) graphs; further work explores sentence-graph retrieval, hierarchical graph indexes, and associative concept graphs (Wu et al., 2025b; Huang et al., 2026; Paul et al., 2026). Most of these systems store memories as entity–relation triples rather than original sentences, losing the nuance and context of the dialogue. These systems therefore cover relations well but semantics and details poorly.

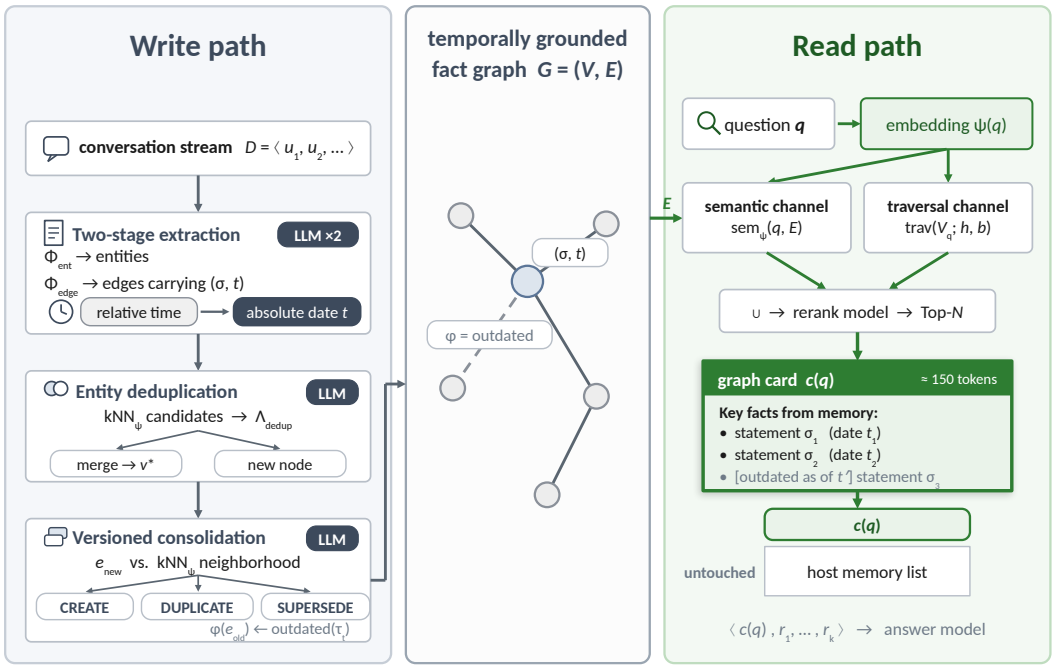


Figure 2: The write and read pipeline of GRAPHCARD. *Write path* (per incoming message $u_t = (s_t, x_t, \tau_t)$): two LLM calls extract entities and candidate edges, each edge carrying a dated, self-contained statement (σ, t), with relative time grounded to absolute dates inside the same extraction call; entities are then resolved by kNN_ψ candidates plus an LLM decision; consolidation compares e_{new} with its neighborhood and creates, discards, or supersedes, where supersession soft-deletes the old edge. *Read path*: one query embedding feeds a semantic channel and a bounded one-hop traversal ($\leq b$ edges per entry entity); pooled candidates are re-ranked by a rerank model, and the top N=10 statements, dates and `[outdated]` marks preserved, render into one $\sim$150-token graph card prepended to the host's untouched list.

2.3 TEMPORAL MECHANISMS

Conversational memory is temporal by nature: facts arrive with timestamps, refer to earlier events, and are later revised or superseded. A memory that cannot tell *when* a fact held, or whether it still holds, fails exactly on temporal and knowledge-update questions. Prior systems each address part of this problem: RecallM (Kynoch et al., 2023) tracks belief updates over a concept graph; Theanine (Ong et al., 2025) organizes memory into temporal chains; Zep/Graphiti (Rasmussen et al., 2025) invalidates outdated edges; TReMu (Ge et al., 2025) builds timeline summaries and offloads date arithmetic to generated code; Nemori (Ma et al., 2026) grounds episodes in absolute time, and SodaMem (Wan et al., 2026) grounds supersede/contradict relations in evidence. Like these systems, we place time at the center of memory construction, resolving relative dates into absolute ones and soft-deleting superseded facts. Moreover, our temporal stack lives in a plug-in graph memory module, so it integrates readily with an existing host and strengthens the host's temporal understanding. In contrast, prior mechanisms are bound to their own systems and hard to integrate elsewhere.

3 METHOD: THE GRAPHCARD PLUG-IN

3.1 PROBLEM SETUP AND NOTATION

A multi-session conversation is a stream $\mathcal{D} = \langle u_1, u_2, \ldots \rangle$, where each message $u_t = (s_t, x_t, \tau_t)$ carries a speaker s_t, text x_t, and a timestamp τ_t. A *host* memory system $\mathcal{H}$ ingests $\mathcal{D}$ into its own store and, given a question q, returns a ranked evidence list $\mathcal{R}_{\mathcal{H}}(q) = \langle r_1, \ldots, r_k \rangle$ under a retrieval budget k. We treat $\mathcal{H}$ as a black box.

The plug-in maintains, independently of $\mathcal{H}$, a temporally grounded fact graph

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}), \qquad e = (v_{\text{subj}},\, v_{\text{obj}},\, \sigma,\, t,\, \rho,\, \phi) \in \mathcal{E},$$

where $\mathcal{V}$ is a set of entities and each edge e is a *reified triple*: subject and object entities $v_{\text{subj}}, v_{\text{obj}} \in \mathcal{V}$, a self-contained natural-language statement σ, an absolute date t, provenance ρ (the source message), and a validity state $\phi \in \{\text{ACTIVE}\} \cup \{\text{OUTDATED}(t') : t'\}$. The statement σ itself is the memory: a self-contained sentence that any answer model can read directly, with no knowledge of the graph's schema. The remaining components support search and maintenance: the entity endpoints anchor traversal, t orders facts in time, ρ records the source, and ϕ marks whether the fact still holds.

At search time, one $\sim$150-token *graph card* $c(q)$ is prepended to the host's list:

$$\mathcal{R}_{\mathcal{H}+\text{card}}(q) \;=\; \langle\, c(q),\, r_1, \ldots, r_k \,\rangle.$$

This design has two defining properties. *Input decoupling*: $\mathcal{G}$ is built from $\mathcal{D}$ alone, never from $\mathcal{H}$'s internal entries. *Output attributability*: the host's retrieval pipeline runs unmodified and its items $r_1, \ldots, r_k$ keep exactly the host's own ranking. The card is prepended as one additional item, so any change in the results can be attributed to exactly that card, which keeps retrieval analysis and evaluation simple. The card does add tokens to the answer context ($\sim$150 per question), but this is a highly favorable trade-off (discussed in Section 4.4): those tokens buy far more accuracy than the same budget spent on additional host entries.

3.2 Incremental graph construction

What the card can carry at search time is bounded by what the graph stores, so most of the design effort goes into the memory write path. A background process consumes one message at a time. Each incoming message is distilled into candidate entities and edges, with relative time resolved to absolute dates. Duplicate entities are merged into existing nodes. Each fact is then consolidated against the store, superseding stale versions. Figure 2 sketches this write path.

Two-stage extraction. Similar to Mem0 (Chhikara et al., 2025), we read each incoming message u_t together with a reference-only sliding window C_t of the preceding w=10 messages. A first LLM call extracts entities (vertices) and a second extracts candidate memories (edges):

$$\hat{\mathcal{V}}_t = \Phi_{\text{ent}}(u_t, C_t), \qquad \hat{\mathcal{E}}_t = \Phi_{\text{edge}}(u_t, C_t, \hat{\mathcal{V}}_t;\, \tau_t).$$

Φ_{edge} is instructed to make each statement σ self-contained and exhaustive over the message's factual content, while preserving the speaker's original wording for subjective claims.

Entity deduplication. The same person or place recurs under many names. If these variants are not merged, $\mathcal{G}$ swells with duplicate nodes, and the facts about one entity scatter across them. To avoid this, each proposed entity $\hat{v}$ is resolved in two steps: a search finds its nearest existing entities, and unless one matches its name exactly, an LLM makes the merge decision,

$$\text{cand}(\hat{v}) = \underset{\psi}{\text{kNN}}\big(\hat{v}, \mathcal{V}\big), \qquad v^* = \Lambda_{\text{dedup}}\big(\hat{v}, \text{cand}(\hat{v})\big) \in \text{cand}(\hat{v}) \cup \{\hat{v}\},$$

where kNN_ψ, used for every lookup in the plug-in, fuses dense retrieval under the embedding ψ with BM25, and Λ_{dedup} either merges $\hat{v}$ into an existing entity (which then accumulates all of its edges) or admits it as new. This keeps the entity set compact as the conversation grows: on LoCoMo, the LLM decision leaves 20% fewer entity nodes than exact-name matching alone (13,190 vs. 16,479 in the Luna group; 11,741 vs. 14,520 in the Haiku group) for the same statements.

Temporal fact management. Relative time expressions inside statement σ (*e.g.*, "yesterday", "last month") are resolved into absolute dates inside the same extraction call, which is why Φ_{edge} takes the message timestamp τ_t as an explicit anchor: every statement σ it emits already carries absolute dates. After ingestion, no stored memory requires date arithmetic to interpret, which is what helps the answer model on temporal questions. Each surviving edge e_{new} is then compared against its semantic neighborhood $N(e_{\text{new}}) = \text{kNN}_\psi(\sigma_{\text{new}}, \mathcal{E})$, and an LLM operator decides among CREATE, DUPLICATE (discard the incoming edge), and SUPERSEDE(e_{old}). Following the edge-invalidation practice of temporal graph memories such as Zep/Graphiti (Rasmussen et al., 2025), supersession is a *soft* deletion:

$$\phi(e_{\text{old}}) \leftarrow \text{OUTDATED}(\tau_t), \qquad \sigma(e_{\text{old}}) \leftarrow \texttt{[outdated as of } \tau_t\texttt{]} \oplus \sigma(e_{\text{old}}),$$

so both versions stay retrievable and stale memories are explicit to the answerer. Rendering the validity state *into the statement text*, rather than as a metadata field, requires no schema support from the host.

Table 1: Accuracy on LoCoMo with different memory configurations (top-10), by GPT-5.6 Luna and Claude Haiku 4.5.

GPT-5.6 Luna					
Method	Single-hop	Multi-hop	Temporal	Open-dom.	Overall
GRAPHCARD-alone	83.5	85.5	84.7	67.7	83.1
Graph-centric memory					
Mem0^g	71.9	74.5	55.1	55.2	67.9
Graphiti	74.7	75.2	72.0	59.4	73.2
A-MEM	71.2	68.4	68.8	63.5	69.7
Vector-based hosts, without and with the plug-in					
RAG (chunk retrieval)	81.8	63.5	81.6	61.5	77.1
+GRAPHCARD	91.3	87.6	91.9	69.8	**89.4** (+12.3)
Nemori	90.2	88.7	79.4	66.7	86.2
+GRAPHCARD	94.1	90.4	93.8	70.8	**91.9** (+5.7)
SimpleMem	89.9	88.3	93.1	68.8	89.0
+GRAPHCARD	93.7	91.8	94.1	76.0	**92.3** (+3.3)
Mem0	92.4	84.8	92.5	68.8	89.5
+GRAPHCARD	93.1	90.1	94.7	72.9	**91.6** (+2.1)
Reference: Full context	96.2	96.5	96.0	77.1	95.0

Claude Haiku 4.5					
Method	Single-hop	Multi-hop	Temporal	Open-dom.	Overall
GRAPHCARD-alone	73.6	80.5	73.5	63.5	74.2
Graph-centric memory					
Mem0^g	48.4	46.5	43.3	49.0	47.0
Graphiti	68.5	73.8	62.6	66.7	68.1
A-MEM	75.7	75.5	72.3	61.5	74.1
Vector-based hosts, without and with the plug-in					
RAG (chunk retrieval)	78.8	64.5	80.1	63.5	75.5
+GRAPHCARD	88.1	86.9	88.2	66.7	**86.6** (+11.1)
Nemori	87.6	88.3	70.1	67.7	82.9
+GRAPHCARD	90.2	93.3	84.4	70.8	**88.4** (+5.5)
SimpleMem	85.6	83.0	87.9	67.7	84.5
+GRAPHCARD	90.5	91.8	90.0	76.0	**89.7** (+5.2)
Mem0	87.5	83.3	90.3	76.0	86.6
+GRAPHCARD	90.1	89.4	91.6	75.0	**89.4** (+2.8)
Reference: Full context	96.4	95.0	93.1	75.0	94.2

3.3 SEARCH-TIME GRAPH CARD

Given a question q, the plug-in gathers candidate facts through two channels and compresses them into one $\sim$150-token card:

$$\mathcal{V}_q = \mathrm{kNN}_\psi(q, \mathcal{V}), \qquad \mathcal{F}(q) \;=\; \mathrm{TopN}\Big(\mathrm{sem}_\psi(q, \mathcal{E}) \;\cup\; \mathrm{trav}\big(\mathcal{V}_q;\, h, b\big)\Big),$$

where sem_ψ is the same hybrid search over statements, and trav expands $\mathcal{V}_q$, the top-5 entities matched in the entity index, to their neighbors. Traversal is bounded: real conversation graphs are heavy-tailed around hubs such as the speakers, so each entry entity contributes at most b edges per hop to keep the expansion tractable. Candidates from both channels are pooled and re-ranked against q by a rerank model, and the card $c(q) = \mathrm{render}(\mathcal{F}(q))$ formats the top $N{=}10$ statements, dates included, under a fixed header, yielding $\sim$150 tokens. Superseded statements keep their `[outdated as of X]` prefix inside the card, so the version chain survives.

Notably, this graph retrieval uses no LLM for entity extraction or query rewriting. The question is embedded once, and this embedding, with BM25 over the question text, drives both channels: it matches statements and locates the traversal entry points in the entity index. Graph retrieval therefore stays fast, and we find that the graph structure and entity retrieval are robust enough for this simple retrieval mechanism.

4 EXPERIMENTS

4.1 SETUP

Benchmarks. *LoCoMo* (Maharana et al., 2024): ten long multi-session conversations. We evaluate all 1,540 scored questions across four categories (single-hop 841, multi-hop 282, temporal 321, open-domain 96), ingesting turn-by-turn (the benchmark's native granularity). *LongMemEval-S* (Wu et al., 2025a): all 500 questions over six types, each spanning ∼50 haystack sessions, ingested as user–assistant pairs. Both benchmarks are evaluated through the `memory-benchmarks` framework (Mem0, 2025), which supplies the dataset loaders, answer prompts, and LLM-judge prompts shared by all systems.

Protocol. We run two end-to-end model groups. Within each group a single model performs ingestion, answering, and judging (*i.e.*, GPT-5.6 Luna (OpenAI, 2026), Claude Haiku 4.5 (Anthropic, 2025)). All systems share the same embedding model (Titan Text Embeddings V2 (Amazon Web Services, 2024), 1024-dim) and the same ingestion granularity, and GRAPHCARD re-ranks its candidates with Cohere Rerank 3.5 (Cohere, 2024). Every reported system is re-run under this shared protocol rather than quoted from its paper. The main retrieval budget is top-10 (for host and GRAPHCARD), and +GRAPHCARD rows add the card on top of the host's top-k list. Full context (the entire conversation in the prompt) serves as a no-retrieval reference.

Systems. Vector-based hosts: chunk RAG (Lewis et al., 2020) (hybrid BM25+dense embeddings over raw chunks), Nemori (Ma et al., 2026), SimpleMem (Liu et al., 2026), and Mem0 (Chhikara et al., 2025). Graph-centric systems: Mem0^g (Chhikara et al., 2025), Graphiti (Rasmussen et al., 2025), and A-MEM (Xu et al., 2026) (classified as graph-centric because its note links participate in retrieval). GRAPHCARD-alone: the retrieved graph card as the solo item. Hosts+GRAPHCARD: the same hosts with the card prepended.

4.2 MAIN RESULTS ON LOCOMO

Table 1 presents the main comparison, one block per model group. Four observations follow.

GRAPHCARD improves every host configuration. All 8 host–model deltas are positive. With the host's pipeline and ranking unchanged, the gains range from +2.1 points on Mem0, already the strongest baseline, to +12.3 on chunk RAG. Notably, Nemori and SimpleMem already carry per-entry temporal mechanisms, yet the card still adds structure that these mechanisms miss: +5.7 on Nemori and +3.3 on SimpleMem (Luna group).

The gains concentrate where relational structure matters. Multi-hop questions improve the most on average: +24.1 on RAG and +1.7 to +5.3 on the memory hosts in the Luna group, +22.4 and +5.0 to +8.8 in the Haiku group. Temporal questions improve on all four hosts. SimpleMem attaches per-entry timestamps, yet the card still lifts its temporal accuracy (+1.0 on Luna, +2.1 on Haiku). Nemori even resolves relative time at write time, as our plug-in does, yet its temporal accuracy is only 79.4 on Luna, far below its single-hop and multi-hop scores, and the card lifts it by +14.4 to 93.8 (+14.3 on Haiku). Knowing *when each fact happened* is not the same as knowing *how facts relate over time*. The card provides the latter by placing the dated facts side by side.

How the graph is added matters more than whether it is added. Graph-centric systems score 67.9 to 73.2 on Luna, far below plain Mem0's 89.5. The difference lies in what the answerer receives. Replacement designs give it short extracted facts *instead of* the host's higher-coverage evidence. GRAPHCARD adds condensed structure *alongside* that evidence, at the cost of only ∼150 extra tokens.

GRAPHCARD and the host are complementary, not redundant. GRAPHCARD-alone scores 83.1/74.2 (Luna/Haiku), below almost every host, yet prepending this same card improves every host. The host supplies coverage and GRAPHCARD supplies structure.

4.3 MAIN RESULTS ON LONGMEMEVAL

LongMemEval evaluates memory over a much longer horizon than LoCoMo: each question is answered against ∼50 haystack sessions. As shown in Table 2, GRAPHCARD still delivers clear gains:

Table 2: Accuracy on LongMemEval-S with different memory configurations (top-10), by GPT-5.6 Luna and Claude Haiku 4.5. Question types: SU/SA/SP = single-session user/assistant/preference, MS = multi-session, KU = knowledge update, TR = temporal reasoning.

GPT-5.6 Luna							
Method	SU	SA	SP	MS	KU	TR	Overall
GRAPHCARD-alone	77.1	62.5	83.3	30.8	55.1	30.8	47.8
Graph-centric memory							
Mem0^g	92.9	12.5	76.7	72.9	74.4	75.9	70.2
Graphiti	38.6	53.6	56.7	12.8	32.1	19.5	28.4
A-MEM	91.4	100.0	93.3	65.4	93.6	71.4	80.6
Vector-based hosts, without and with the plug-in							
RAG (chunk retrieval)	95.7	100.0	93.3	57.9	94.9	75.9	80.6
+GRAPHCARD	97.1	100.0	86.7	63.2	89.7	78.2	**81.6** (+1.0)
Nemori	94.3	94.6	93.3	71.4	82.1	75.2	81.2
+GRAPHCARD	97.1	96.4	93.3	78.9	88.5	78.9	**85.8** (+4.6)
SimpleMem	87.1	87.5	90.0	68.4	94.9	75.2	80.4
+GRAPHCARD	91.4	92.9	93.3	71.4	92.3	79.7	**83.4** (+3.0)
Mem0	88.6	87.5	83.3	65.4	93.6	73.7	78.8
+GRAPHCARD	90.0	89.3	90.0	68.4	85.9	77.4	**80.2** (+1.4)
Reference: Full context	97.1	100.0	90.0	84.2	96.2	90.2	91.6
Claude Haiku 4.5							
Method	SU	SA	SP	MS	KU	TR	Overall
GRAPHCARD-alone	35.7	55.4	36.7	13.5	47.4	18.0	29.2
Graph-centric memory							
Mem0^g	90.0	14.3	60.0	54.9	74.4	65.4	61.4
Graphiti	32.9	53.6	33.3	13.5	23.1	18.0	24.6
A-MEM	84.3	96.4	80.0	57.9	85.9	72.2	75.4
Vector-based hosts, without and with the plug-in							
RAG (chunk retrieval)	92.9	98.2	76.7	55.6	92.3	74.4	77.6
+GRAPHCARD	90.0	96.4	70.0	61.7	93.6	75.2	**78.6** (+1.0)
Nemori	82.9	75.0	76.7	72.9	83.3	76.7	77.4
+GRAPHCARD	91.4	82.1	83.3	76.7	88.5	79.7	**82.4** (+5.0)
SimpleMem	81.4	76.8	70.0	57.9	84.6	70.7	71.6
+GRAPHCARD	85.7	82.1	70.0	62.4	85.9	75.9	**75.6** (+4.0)
Mem0	91.4	78.6	80.0	55.6	88.5	74.4	74.8
+GRAPHCARD	85.7	82.1	73.3	60.2	92.3	79.7	**77.2** (+2.4)
Reference: Full context	90.0	94.6	53.3	70.7	78.2	69.9	76.0

all 8 host–model combinations improve (+1.0 to +5.0). The gains again concentrate on multi-session and temporal-reasoning questions, which require evidence spanning multiple sessions. The largest gains are on Nemori (+4.6 on Luna, +5.0 on Haiku), and Nemori+GRAPHCARD at 85.8 is the strongest retrieval configuration on this benchmark. Notably, on the Haiku group, full context drops to 76.0, below the naive RAG baseline, while three host+GRAPHCARD configurations beat it by up to 6.4 points (77.2, 78.6, and 82.4). This shows that simply stacking the whole dialogue history into the prompt does not reliably scale to extremely long conversations (*i.e.*, >100K tokens) and can severely hurt answer quality.

4.4 TOKEN EFFICIENCY OF GRAPHCARD

Figure 3 traces the accuracy–token frontier of Mem0 (Chhikara et al., 2025), the strongest host on LoCoMo, under the two ways of buying more context: retrieving more host entries, or rendering more card statements. Gray traces Mem0 alone from top-10 to top-50; green widens the host to top-k while keeping the standard 10-statement GRAPHCARD; purple holds the host at top-10 and enlarges the GRAPHCARD to 50 statements, and the star marks the full-context reference. Both routes start from the same configuration, Mem0 at top-10, which scores 89.5 on a median 381 memory tokens

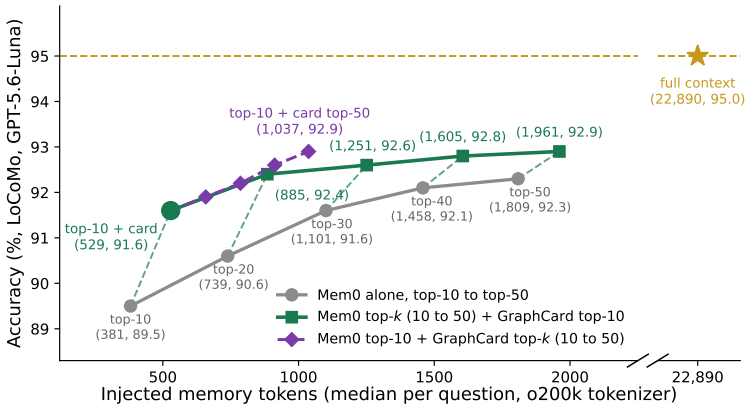


Figure 3: Accuracy vs. injected memory tokens on LoCoMo (Luna group).

Table 3: Effect of write-time temporal grounding on RAG+GRAPHCARD (LoCoMo, Luna group).

Configuration	Single-hop	Multi-hop	Temporal	Open-dom.	Overall
w/ temporal grounding (full pipeline)	91.3	87.6	91.9	69.8	89.4
w/o temporal grounding	89.7	85.1	85.4	65.6	86.4 (−3.0)

per question, so what separates them is the increment each spends beyond it. Reaching 91.6 raises that median by 148 tokens with the card, against 720 by widening the host to its own top-30: the same accuracy with 79% fewer additional tokens. Read the other way, at a matched budget the card route stays about 1.5 points above the host's own curve (+1.47 to +1.65 over the range it spans). GRAPHCARD also reaches accuracies the host cannot buy even at top-50: wider retrieval saturates at 92.3, gaining only 0.2 over its final step, whereas the card climbs to 92.9 with the host still at top-10 and 50 statements rendered. These results show that memory tokens spent on the card buy more accuracy than the same tokens spent on wider host retrieval.

4.5 EFFECTIVENESS OF TEMPORAL GROUNDING

Table 3 measures the effect of write-time temporal grounding. The ablated variant stores relative time expressions verbatim ("yesterday" stays "yesterday") and falls back to the message timestamp for edge dates. The graph is rebuilt from scratch and the card regenerated, while the host side stays frozen with the identical RAG retrieval of Table 1, so the only variable is the card. Removing grounding costs 3.0 points overall, and the largest drop is on temporal questions (−6.5): without absolute dates, every temporal answer requires date arithmetic at read time.

5 CONCLUSION

We presented GRAPHCARD, a graph memory plug-in for long-term conversational memory. The plug-in builds a temporally grounded fact graph in the background and leaves the host memory untouched. At search time, one LLM-free search, 0.61 s at p50, produces a ∼150-token graph card, prepended to the host's retrieved list. GRAPHCARD improves every host memory system across two benchmarks and two model groups, by up to +12.3 points, and the gains concentrate on multi-hop and temporal questions. It also yields a better accuracy–token trade-off than widening the host's retrieval, matching Mem0's top-30 accuracy on LoCoMo with 79% fewer additional tokens. GRAPHCARD-alone scores below almost every host yet improves every host it is added to, showing that graph structure and vector coverage are complements, not substitutes. More broadly, this suggests a general paradigm for bringing graph signals into memory systems: the graph enters as a plug-in beside the existing memory, rather than as its replacement.

REPRODUCIBILITY STATEMENT

All systems in this paper are re-run under one shared protocol rather than quoted from their papers; Section 4.1 specifies the benchmarks, models, embedding model, ingestion granularity, and retrieval budgets. To support reproduction, Appendix A gives the complete write and read paths in pseudocode, Appendix B lists every plug-in hyperparameter, and Appendix F reproduces all four LLM prompts. Appendix C reports paired bootstrap confidence intervals and exact McNemar tests for all 16 primary comparisons. Both benchmarks (LoCoMo, LongMemEval) and the evaluation framework are publicly available.

AI USE STATEMENT

In this work, we did not use generative AI tools for any task with required disclosure: we did not use them to propose or refine hypotheses, design or provide feedback on the methodology or experiments, implement methods, generate synthetic datasets, clean or reformat datasets, assist with translation, or interpret results, and the remaining tasks with required disclosure (theoretical models, mathematical claims, proofs, and qualitative or thematic data analysis) are not applicable to this work. Additionally, we used generative AI tools for tasks with recommended disclosure: editing the paper to improve readability, brainstorming the design of the illustrative figures, and proposing the system name GRAPHCARD. We have reviewed all AI-assisted work: all AI-edited text was reviewed and finalized by the authors, and all figures were created by the authors following the design discussions. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI. Separately from these uses, LLMs are a component of the studied system itself: the proposed plug-in invokes LLMs for extraction, deduplication, and consolidation (Section 3), and LLMs serve as the ingestion, answering, and judging models in the evaluation (Section 4.1).

REFERENCES

Amazon Web Services. Amazon Titan Text Embeddings V2. `https://docs.aws.amazon.com/bedrock/latest/userguide/titan-embedding-models.html`, 2024.

Anthropic. Claude Haiku 4.5. `https://www.anthropic.com/news/claude-haiku-4-5`, 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.

Cohere. Cohere Rerank 3.5. `https://cohere.com/blog/rerank-3pt5`, 2024.

Yubin Ge, Salvatore Romeo, Jason Cai, Raphael Shu, Yassine Benajiba, Monica Sunkara, and Yi Zhang. Tremu: Towards neuro-symbolic temporal reasoning for llm-agents with memory in multi-session dialogues. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 18974–18988, 2025.

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.

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.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Brandon Kynoch, Hugo Latapie, and Dwane van der Sluis. Recallm: An adaptable memory mechanism with temporal understanding for large language models. *arXiv preprint arXiv:2307.02738*, 2023.

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.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. In *Forty-third International Conference on Machine Learning*, 2026.

Wenquan Ma, Jiayan Nan, and Wenlong Wu. What deserves memory: Adaptive memory distillation for llm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34789–34812, 2026.

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.

Mem0. memory-benchmarks: Evaluation suite for long-term conversational memory. `https://github.com/mem0ai/memory-benchmarks`, 2025.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seungwon Hwang, Dongha Lee, and Jinyoung Yeo. Towards lifelong dialogue agents via timeline-based memory management. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8631–8661, 2025.

OpenAI. GPT-5.6. `https://openai.com/index/gpt-5-6/`, 2026.

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, et al. Secom: On memory construction and retrieval for personalized conversational agents. In *International Conference on Learning Representations*, volume 2025, pp. 91851–91885, 2025.

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.

Daivik Patel and Shrenik Patel. Engram: Effective, lightweight memory orchestration for conversational agents. *arXiv preprint arXiv:2511.12960*, 2025.

Swarna Kamal Paul, Shubhendu Sharma, and Nitin Sareen. GAAMA: Graph augmented associative memory for agents. *arXiv preprint arXiv:2603.27910*, 2026.

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.

Fengrong Wan, Chengcan Wu, and Ningtao Lyu. SodaMem: Evidence-grounded temporal graph memory for LLM agents. *arXiv preprint arXiv:2608.08055*, 2026.

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.

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

Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 5180–5197, 2022a.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Xinchao Xu, Zhibin Gou, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, and Shihang Wang. Long time no see! open-domain conversation with long-term persona memory. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2639–2650, 2022b.

Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, et al. Graph-based agent memory: Taxonomy, techniques, and applications. *arXiv preprint arXiv:2602.05665*, 2026.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 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.

A PSEUDOCODE

Algorithms 1 and 2 give the complete plug-in in pseudocode. The write path runs in the background per incoming message; the read path runs at question time.

Algorithm 1 Write path: ingest one message

Require: message u_t with timestamp τ_t, sliding context C_t, graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

$\hat{\mathcal{V}} \leftarrow \Phi_{\text{ent}}(u_t, C_t)$ ▷ propose entity mentions (LLM)
$\hat{\mathcal{E}} \leftarrow \Phi_{\text{edge}}(u_t, C_t, \hat{\mathcal{V}};\, \tau_t)$ ▷ self-contained σ, dates grounded to τ_t (LLM)
for all $\hat{v} \in \hat{\mathcal{V}}$ **do** ▷ entity deduplication
  $\text{cand} \leftarrow \text{kNN}_{\psi}(\hat{v}, \mathcal{V})$ ▷ retrieve entity candidates
  $v^* \leftarrow \Lambda_{\text{dedup}}(\hat{v}, \text{cand})$ ▷ merge into existing entity, or admit as new
end for
for all $\hat{e} \in \hat{\mathcal{E}}$ **do**
  $a \leftarrow \Lambda_{\text{consolidate}}\big(\hat{e},\, \text{kNN}_{\psi}(\sigma(\hat{e}), \mathcal{E})\big)$ ▷ CREATE / DUPLICATE / SUPERSEDE(e_{old})
  if $a = \text{SUPERSEDE}(e_{\text{old}})$ **then**
    $\sigma(e_{\text{old}}) \leftarrow$ `[outdated as of` τ_t`]` $\oplus\, \sigma(e_{\text{old}})$ ▷ soft delete
  end if
  if $a \neq$ DUPLICATE **then** $\mathcal{E} \leftarrow \mathcal{E} \cup \{\hat{e}\}$
  end if
end for

Algorithm 2 Read path: build the graph card and compose with the host memory list

Require: question q, graph $\mathcal{G}$, host list $\mathcal{R}_{\mathcal{H}}(q) = \langle r_1, \ldots, r_k \rangle$, card size $N{=}10$

$\mathcal{V}_q \leftarrow \text{kNN}_{\psi}(q, \mathcal{V})$ ▷ top-5 entry entities by hybrid match
$\mathcal{E}_{\text{trav}} \leftarrow \text{expand}(\mathcal{V}_q;\, h{=}1,\, \text{cap } b \text{ per entity})$ ▷ bounded traversal
$\mathcal{E}_{\text{sem}} \leftarrow \text{sem}_{\psi}(q, \mathcal{E})$ ▷ hybrid semantic channel, top-50
$\mathcal{F} \leftarrow \text{TopN}\,\big(\text{rerank}(q,\, \mathcal{E}_{\text{sem}} \cup \mathcal{E}_{\text{trav}})\big)$ ▷ rerank model
$c(q) \leftarrow \text{render}($`"Key facts from memory:"`$,\, \mathcal{F})$ ▷ the ∼150-token graph card
return $\langle\, c(q),\, r_1, \ldots, r_k \,\rangle$ ▷ GRAPHCARD + memory list

B PLUG-IN HYPERPARAMETERS

Table 4: Plug-in hyperparameters used in all experiments.

Stage	Parameter	Value
Extraction	sliding context window w	10 messages
Deduplication	candidate entities per new entity	top-5
Consolidation	semantic neighbors compared	top-10
Semantic channel	statements retrieved by hybrid search	top-50
Traversal	entry entities matched per query	top-5
Traversal	hops h	1
Traversal	edge cap b per entry entity and hop	500
Card	statements rendered N	10

Hybrid search. The plug-in performs nearest-neighbor lookups in four places: candidate entities for deduplication, neighbor statements for consolidation, the semantic channel, and entry-entity matching. All four use the same hybrid search. Given a query text and its embedding, dense kNN and BM25 each retrieve twice the requested number of candidates (*e.g.*, 100 each for the top-50 semantic channel). Each score list is min-max normalized to $[0, 1]$, the two scores are averaged with equal weights, and the top-ranked results are returned.

C STATISTICAL ROBUSTNESS OF THE PRIMARY COMPARISONS

Table 5 reports question-level uncertainty for all 16 primary comparisons. For each host-versus-host+GRAPHCARD pair we align the two runs question by question and report a paired bootstrap 95% confidence interval (10,000 resamples) and the exact two-sided McNemar p-value.

All 8 LoCoMo gains are individually significant: every confidence interval excludes zero and every $p \leq 0.002$. On LongMemEval, 4 of 8 comparisons are significant at $p < 0.05$; the 4 smallest gains (RAG and Mem0) are positive, but with only 500 questions their confidence intervals still include zero. The zero crossings come from the small question set, and the intervals lean heavily to the positive side: for Mem0 on Haiku the interval reaches only 0.4 points below zero but 5.2 points above it, around an observed delta of $+2.4$, and the other 3 cases show the same shape. These 4 gains also point in the same direction as the other 12 comparisons, all of which are individually significant.

Table 5: Question-level robustness of the 16 primary comparisons. CI: paired bootstrap 95% confidence interval. p: exact McNemar test (two-sided).

Benchmark	**Host**	**Base**	**+GRAPHCARD**	**Δ**	**95% CI**	p
GPT-5.6 Luna group						
LoCoMo	RAG	77.1	89.4	+12.3	[+10.4, +14.2]	2×10^{-37}
LoCoMo	Nemori	86.2	91.9	+5.7	[+4.2, +7.1]	1×10^{-13}
LoCoMo	SimpleMem	89.0	92.3	+3.3	[+2.1, +4.7]	9×10^{-7}
LoCoMo	Mem0	89.5	91.6	+2.1	[+0.8, +3.4]	0.002
LongMemEval	RAG	80.6	81.6	+1.0	[−1.2, +3.4]	0.500
LongMemEval	Nemori	81.2	85.8	+4.6	[+2.0, +7.2]	0.001
LongMemEval	SimpleMem	80.4	83.4	+3.0	[+0.6, +5.4]	0.020
LongMemEval	Mem0	78.8	80.2	+1.4	[−1.0, +3.8]	0.337
Claude Haiku 4.5 group						
LoCoMo	RAG	75.5	86.6	+11.1	[+9.2, +12.9]	5×10^{-31}
LoCoMo	Nemori	82.9	88.4	+5.5	[+4.0, +7.1]	8×10^{-12}
LoCoMo	SimpleMem	84.5	89.7	+5.2	[+3.6, +6.8]	6×10^{-11}
LoCoMo	Mem0	86.6	89.4	+2.8	[+1.2, +4.3]	4×10^{-4}
LongMemEval	RAG	77.6	78.6	+1.0	[−1.8, +4.0]	0.590
LongMemEval	Nemori	77.4	82.4	+5.0	[+2.0, +8.0]	0.001
LongMemEval	SimpleMem	71.6	75.6	+4.0	[+1.2, +6.8]	0.009
LongMemEval	Mem0	74.8	77.2	+2.4	[−0.4, +5.2]	0.141

D INJECTED MEMORY TOKENS

Figure 4 shows the per-question size of the graph card on LoCoMo when it renders the top-N statements for N from 10 (the setting used throughout the paper) to 50, in both model groups. Token counts for the GPT-5.6 Luna group use OpenAI's `o200k_base` tokenizer[1]; counts for the Claude Haiku 4.5 group use the Bedrock CountTokens API for Claude[2]. At N=10 the median card is 144 tokens in the Luna group and 176 in the Haiku group. The card grows close to linearly with N, reaching a median of 655 and 791 tokens at N=50.

E LLM ENTITY EXTRACTION FOR SEARCH

The plug-in's read path locates traversal entry points by matching the question text directly against the entity index, with no LLM call. A heavier variant first analyzes the question with one LLM call to extract entities. Table 6 compares the two end-to-end under identical conditions. The LLM-assisted variant nearly triples median latency for no measurable accuracy gain. Direct matching is therefore the only mode used in this paper.

[1] `https://github.com/openai/tiktoken`

[2] `https://docs.aws.amazon.com/bedrock/latest/userguide/count-tokens.html`

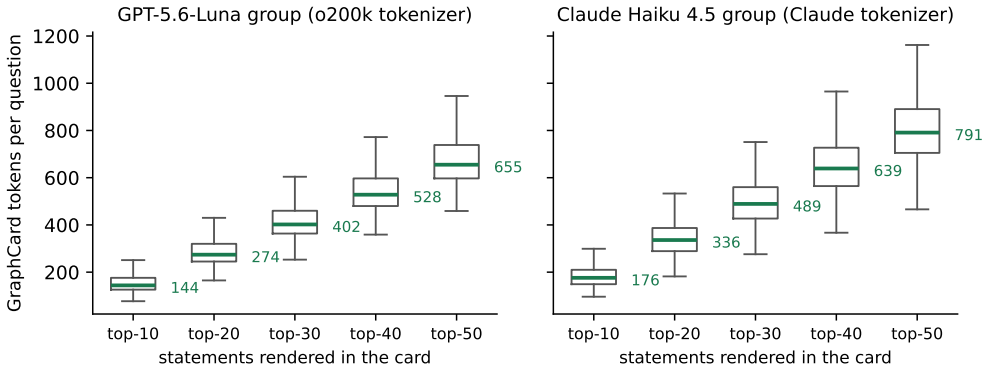


Figure 4: Graph card size in tokens per question on LoCoMo. Left: GPT-5.6 Luna group, `o200k_base` tokenizer. Right: Claude Haiku 4.5 group, Claude tokenizer.

Table 6: The card's two query-analysis modes under identical conditions: search latency vs. end-to-end accuracy (LoCoMo, Luna group, Mem0+GRAPHCARD at top-10).

Query-analysis mode	**Search p50**	**Search p95**	**Overall**
Direct query matching (default)	0.61 s	0.89 s	91.6
w/ LLM entity extraction	1.68 s	2.76 s	91.5

F PROMPTS

F.1 ENTITY EXTRACTION (Φ_{ent})

System prompt:

```
You are an expert entity extraction system. Extract entities from the
conversation.

Extract entities that are:
- People (names, roles, relationships)
- Places (locations, addresses, venues)
- Organizations (companies, institutions, groups)
- Concepts (ideas, topics, subjects)
- Events (meetings, activities, occasions)
- Objects (items, products, things)

The conversation took place on [CONVERSATION DATE, e.g. Friday, May 08,
 2020].
- Also extract as concept entities: notable attributes, qualities,
feelings and descriptive words that facts will attach to (e.g. "
tremendous", "vegetarian dishes"), so that relationships about them can
 be created.
- BE EXHAUSTIVE: cover every person, place, activity, object, plan and
quality mentioned.

Special handling for pronouns:
- If the message content begins with a speaker name (e.g. "Name: ..."),
 attribute all self references ('I', 'me', 'my') in that message to
THAT SPEAKER by name -- never to "the user"
- Only if no speaker name is present: USER self references map to "the
user"; ASSISTANT self references are IGNORED

Return a JSON array of entities with this exact format:
```json
[
```
```

```
  {
    "name": "entity name",
    "type": "person|place|organization|concept|event|object"
  }
]
```

Only extract entities that are explicitly mentioned. Do not infer or assume entities.
Return empty array [] if no entities found.

User message:

```
## Previous messages (context only - resolve references against these, but do NOT extract facts from them)
[PRECEDING w=10 MESSAGES, each rendered as "Role:" followed by its text]

## New messages (extract entities from these)
[TARGET MESSAGE TEXT]
```

F.2 EDGE EXTRACTION (Φ_{edge})

System prompt:

```
You are an expert relationship extraction system. Extract relationships between entities from the conversation.

Entities found in the message: [COMMA-SEPARATED ENTITY NAMES FROM THE FIRST CALL]

Extract relationships that connect these entities. Create natural language statements in subject-predicate-object format.

The conversation below took place on [CONVERSATION DATE, e.g. Friday, May 08, 2020]. Follow these rules:
- Resolve ALL relative time expressions to absolute dates in the statement text: "yesterday" -> the actual date, "last year" -> the actual year, "next month" -> the actual month and year. Never leave words like "yesterday", "recently", "last week" unresolved in a statement.
- When the conversation states or implies WHEN an event happened or will happen, include that absolute date (or month + year) in the statement, e.g. "Priya took up falconry in March 2021" rather than "Priya took up falconry recently".
- Distinguish the EVENT time from the CONVERSATION time: only attach a date to an event if the conversation says when it actually happened. Never assume an event happened on the conversation date just because it was mentioned that day (mentioning an event today does not mean it happened today).
- When a speaker states a duration or anniversary ("we have known each other for six years"), record the duration itself in the statement, resolved against the conversation date (e.g. "As of April 2020, Riley and Casey have known each other for six years (since 2014)").
- Each statement must be SELF-CONTAINED and understandable without the conversation: include the specific object, name, or detail (write "Priya plans to visit the botanical garden in Zurich in autumn 2020", never "Priya is planning a visit").
- PRESERVE the speaker's specific wording for descriptions, feelings and qualities: if a speaker calls a trip "tremendous", the statement must keep the word "tremendous", not a paraphrase like "enjoyable".
```

```
- BE EXHAUSTIVE: every distinct fact mentioned in the conversation (
events, dates, preferences, plans, feelings, possessions, quantities)
should appear in some statement. An attribute or quality may be the
object of a statement (list it as a concept entity if needed). Do not
summarize multiple facts into one vague statement.

Special handling for pronouns in relationships:
- If the message content begins with a speaker name (e.g. "Name: ..."),
 attribute all self references ('I', 'me', 'my') in that message to
THAT SPEAKER by name -- never to "the user"
- Only if no speaker name is present: USER self references map to "the
user"; ASSISTANT self references are IGNORED

Return a JSON array of memories with this exact format:
```json
[
  {
    "statement": "natural language statement describing the
relationship",
    "subject_name": "subject entity name",
    "predicate_name": "relationship/action name",
    "object_name": "object entity name",
    "confidence": 0.95
  }
]
```

Only extract relationships that are explicitly stated or strongly
implied in the message.
Confidence should be between 0.0 and 1.0.
Return empty array [] if no relationships found.
```

User message:

```
## Previous messages (context only - resolve references against these,
but do NOT extract facts from them)
[PRECEDING w=10 MESSAGES, each rendered as "Role:" followed by its text
]

## New messages (extract relationships from these)
[TARGET MESSAGE TEXT]
```

F.3 Entity deduplication (Λ_{dedup})

System prompt:

```
You are an entity deduplication expert. Decide whether a new entity
should reuse an existing similar entity or be created as a new entity.

Consider:
- Semantic similarity of names
- Entity types compatibility
- Avoid over-merging distinct entities

Respond with JSON:
```json
{
    "reuse": true/false,
    "entity_id": "id_if_reuse_or_null",
    "reason": "explanation"
}
```
```

User message:

```
New entity: [NEW ENTITY NAME]

Similar existing entities:
ID: [CANDIDATE ID]:
Name: [CANDIDATE NAME]
Type: [CANDIDATE TYPE]

[... one block per candidate ...]
```

F.4 TEMPORAL CONSOLIDATION

This prompt asks the LLM to decide deletions, but in practice nothing is deleted: the affected statement is kept and marked as outdated instead.

System prompt:

```
You are an AI memory management system. Your task is to analyze a new
memory against existing memories and decide how to update the memory
database.

## Input
- A new memory (i.e., statement)
- A list of existing related memories

## Task
1. DELETE: IF the new memory contradicts existing ones, THEN mark
contradictory existing memories for deletion.
2. CREATE: After considering DELETION, decide if the new memory should
be created:
   - Check if the new memory is redundant with the REMAINING memories (
excluding the ones you marked for deletion)
   - Only create the new memory if it provides unique information not
covered by the remaining memories

## Rule
ONLY delete if ALL these conditions are met:
- The statements are about the EXACT SAME specific entity/person/object
- They make contradictory claims about the SAME attribute
- Both statements cannot be true simultaneously

Valid deletion example:
- "Mireille Dubois is 25 years old" vs "Mireille Dubois is 30 years old
" (same person, contradictory ages)

NEVER delete for examples:
- Different entities: "University of Chicago" vs "Harvard University" (
different schools)
- Different topics: "algorithms" vs "pharmacies" vs "ships" (unrelated
subjects)
- Different aspects: "Mireille proposed a merger" vs "Students are
funded" (different facts about the same organization)
- Compatible facts that can coexist

When ANY doubt exists, keep both memories.

## Output format
```json
{
    "delete_indices": [],
    "create_new": true/false
}
```
```

```
```

User message:

```
## Existing memories
Statement 0: [NEAREST STORED STATEMENT]
Statement 1: [NEXT STORED STATEMENT]
[... up to top-10 ...]

## New memory
[NEW STATEMENT]
```