SPECTRUM OF ENGINEERING SCIENCES
ISSN (E) 3007-3138 (P) 3007-312X VOLUME 4, ISSUE 6, 2026

WHEN DOES GRAPH-STRUCTURED MEMORY HELP MULTI-SESSION LLM AGENTS? AN EMPIRICAL STUDY OF HYGRAM, A HYBRID GRAPH–VECTOR MEMORY ARCHITECTURE

Muhammad Ismail[*1], **Azeem Akram**[2]

[*1]*MS Software Engineering, Riphah International University, Islamabad I-14, Pakistan*
[2]*Master in Software Engineering, Riphah International University, Islamabad I-14, Pakistan*

[*1]mi477048@gmail.com, [2]akramazeem947@gmail.com

DOI: https://doi.org/10.5281/zenodo.20957473

Keywords
LLM agents; long-term memory; knowledge graphs; retrieval-augmented generation; hybrid retrieval; temporal knowledge graphs; multi-session dialogue; hallucination mitigation; persistent contextual reasoning.

Article History
Received: 25 April 2026
Accepted: 04 June 2026
Published: 21 June 2026

Corresponding Author: *
Muhammad Ismail

Abstract

*Large language model (LLM) agents are fundamentally stateless: when a session terminates, the agent loses all episodic context, and subsequent interactions begin from a blank slate. The prevailing remedy stores prior dialogue in a vector database and retrieves semantically similar text chunks at query time. This flat retrieval paradigm discards the relational structure that binds facts together, provides no principled mechanism for distinguishing stale from currently-valid information, and treats memory as a passive evidence store rather than an active, structured representation. A natural hypothesis is that representing memory as a knowledge graph and retrieving it through traversal will improve relationship- and time-dependent reasoning. This paper presents **HyGRAM** (Hybrid Graph Retrieval-Augmented Memory)—which extracts timestamped subject–relation–object triples from each session into a temporally-aware graph, retrieves by seeding with dense vector similarity and expanding through bounded multi-hop traversal, and consolidates the graph so new evidence can invalidate prior beliefs—and **tests that hypothesis empirically** against no-memory, flat vector, and graph-only baselines on the LoCoMo benchmark, using entirely free and open tooling and a commodity open model. The principal result is **negative and instructive**: in this regime the hybrid graph memory did not outperform flat vector retrieval (vector-only achieved the highest accuracy, and the gap persisted on the multi-hop questions graph traversal was expected to favour), and explicit temporal consolidation produced no measurable change in accuracy or contradiction rate. The finding is **robust across two extractor scales**: replicating with a 7B model raised every condition's accuracy but widened the vector baseline's lead (26.0% versus HyGRAM's 8.0%) rather than closing it, and on the two adequately-sampled question categories—multi-hop and temporal—vector retrieval led decisively. We trace the outcome to **extraction**: converting dialogue into triples is lossy, and a more capable model exploits the verbatim text retained by a flat store more effectively than the graph; extraction quality appears necessary but not sufficient for graph memory to pay off at this scale. The contributions are therefore a reproducible architecture and*

pipeline, and a controlled, honestly-reported study that delimits when graph-structured agent memory is likely to help and identifies extraction as the first-order lever.

1. Introduction

Conversational and tool-using agents built on large language models have advanced rapidly, yet they remain bounded by a finite context window and by the absence of any durable state across sessions. Within a single session, the transformer attends over the full transcript; once the session ends, that transcript is gone. An agent that helped a user design a retrieval system on Monday has no recollection of it on Friday unless the relevant text is re-supplied. For assistants intended to accompany a user over weeks or months—coding copilots, tutors, health and productivity companions—this statelessness is not a peripheral inconvenience but a defining limitation.

The dominant engineering response is retrieval-augmented generation (RAG) over a conversational archive: past turns are embedded into a vector space, and at query time the system retrieves the top-k most similar chunks and prepends them to the prompt. This approach is simple, scalable, and often adequate when the answer resides verbatim in a single retrievable passage. Its weaknesses, however, are structural rather than incidental. First, semantic similarity retrieves passages that *resemble* the query, not passages that are *causally or relationally connected* to it; multi-hop questions whose answer depends on chaining several facts are poorly served because the bridging facts may be individually dissimilar to the query. Second, a flat vector store has no native notion of time or validity: a superseded preference ("I use TensorFlow") and its correction ("I switched to PyTorch") coexist as equally retrievable chunks, and the agent has no principled basis for preferring the current one. Third, memory in this paradigm is purely passive—an evidence store consulted on demand—rather than an evolving representation that accumulates and reorganises knowledge as a human's associative memory does.

These limitations motivate a structured alternative. Knowledge graphs encode entities and the typed, directed relationships between them, making relational and multi-hop reasoning a first-class operation (graph traversal) rather than an emergent hope of embedding proximity. Recent systems—GraphRAG for query-focused summarisation, and Zep, Mem0, and A-MEM for agent memory specifically—demonstrate that graph-structured or dynamically-linked memory can outperform flat retrieval on relationship-dependent tasks. Yet the relative contribution of the two retrieval modalities, dense vector search and graph traversal, is rarely isolated, and the role of explicit temporal validity in suppressing contradictions is under-examined.

This paper makes the following contributions:

1. **A hybrid memory architecture, HyGRAM,** that unifies dense vector retrieval and bounded graph traversal over a single temporally-annotated knowledge graph, with temporal consolidation that invalidates superseded facts after each session.
2. **A fully reproducible, zero-cost implementation** built entirely on free, open components (an open model served by Ollama, sentence-transformers embeddings, a NetworkX temporal graph, and an in-process vector index), released so that the central question—whether stronger extraction reverses our result—can be tested directly.
3. **A controlled empirical study** on the LoCoMo benchmark that disentangles vector search, graph traversal, and temporal consolidation, and finds, contrary to the motivating hypothesis, that the hybrid graph memory does not beat flat vector retrieval under a commodity small model.
4. **A diagnosis and an honest negative result:** we attribute the outcome to extraction quality—the binding constraint on graph memory—and argue that practitioners should treat extractor quality, not retrieval sophistication, as the first-order design lever.

The remainder of the paper is organised as follows. Section 2 reviews related work and contrasts existing approaches. Sections 3 and 4 state the

research gap and problem precisely. Section 5 lists objectives. Sections 6–8 detail the methodology, architecture, and experimental setup. Sections 9 and 10 present the results protocol and comparative analysis. Sections 11–13 discuss limitations, future work, and conclusions.

2. Literature Review and Related Work

We organise prior work into four strands: (i) context-extension and summarisation memory, (ii) flat retrieval-augmented memory, (iii) graph-structured and temporally-aware memory, and (iv) evaluation benchmarks and metrics.

2.1 Context-extension and summarisation memory

MemGPT frames the LLM as an operating system that pages information between a bounded context window ("main memory") and external storage, using function calls to decide what to retain, evict, and recall. It introduced the Deep Memory Retrieval (DMR) benchmark and reported 93.4% accuracy with GPT-4-turbo, against a recursive-summarisation baseline of 35.3%, establishing that explicit memory management vastly outperforms naive summarisation. MemoryBank takes a cognitively-motivated route, modulating memory strength with an Ebbinghaus-style forgetting curve so that less-reinforced memories decay. Both demonstrate the value of managed memory, but both retrieve over essentially flat stores and do not represent inter-fact relationships explicitly.

2.2 Flat retrieval-augmented memory

The mainstream pattern embeds dialogue turns and retrieves the top-k by cosine similarity, a direct descendant of open-domain RAG. It is strong on single-hop recall and trivially scalable, but, as Section 1 argued, it is blind to relational structure and to temporal validity. Mem0 advances this line for production settings: it extracts salient facts from each exchange and maintains them through explicit ADD/UPDATE/DELETE/NOOP operations, thereby addressing memory staleness through update operations and reporting large latency and token-cost reductions versus full-context baselines. Mem0 also offers a graph-augmented variant, signalling the field's movement toward structure.

2.3 Graph-structured and temporally-aware memory

GraphRAG constructs an entity–relationship graph from a corpus, detects communities, and produces hierarchical summaries to answer global, query-focused questions that flat RAG cannot. Zep introduces a temporal knowledge-graph engine (Graphiti) for agent memory that maintains validity intervals on edges, allowing the system to reason about when a fact held; it reports surpassing MemGPT on DMR (≈94.8%). A-MEM adopts a Zettelkasten philosophy: each memory is a richly-attributed note that is dynamically linked to related notes through embedding similarity and LLM reasoning, and a "memory evolution" step lets new notes retroactively update the context of older ones. HyGRAM draws on all three—graph structure from GraphRAG, temporal validity from Zep, and evolution/consolidation from A-MEM—but differs in making the *interaction* between dense seeding and bounded traversal an explicit, ablatable design axis rather than committing wholesale to either modality.

2.4 Benchmarks and evaluation

DMR (introduced with MemGPT) comprises 500 multi-session conversations of five sessions each, with up to twelve messages per session and a question/answer probe per conversation; it has become a standard but is now near-saturated for strong systems and emphasises single-fact recall. LoCoMo (Maharana et al., 2024) is a more demanding benchmark of 50 very long human-collected conversations averaging 19.3 sessions and roughly 300 turns (~9,200 tokens) each, with 7,512 QA pairs spanning single-hop, multi-hop, temporal, open-domain, and adversarial categories—precisely the relationship- and time-dependent reasoning that motivates graph memory. HotpotQA provides multi-hop questions with supporting-fact supervision and is a useful diagnostic for traversal-dependent reasoning, though it is not multi-session. RAGAS supplies reference-free metrics (faithfulness, answer relevancy, context precision and recall) for the

retrieval and generation stages. Table 1 summarises the comparison.

Table 1. Critical comparison of representative agent-memory approaches.

System / Approach	Memory representation	Temporal validity	Multi-hop retrieval	Key limitation
No-memory / full-context	None / raw transcript	No	Within window only	No persistence; context blow-up and cost
MemGPT	Paged flat store	Implicit (recency)	Limited	No explicit relations; manual paging policy
MemoryBank	Flat + decay curve	Decay heuristic	No	Forgetting is heuristic, not relational
Vector-only RAG	Dense embeddings	No	Weak (similarity only)	Blind to relations and staleness
Mem0	Extracted facts + ops	Via UPDATE/DELETE	Partial (graph variant)	Relations secondary to fact list
GraphRAG	Entity graph + communities	No	Strong (global)	Corpus-oriented, not session memory; costly indexing
Zep	Temporal KG (Graphiti)	Yes (edge intervals)	Strong	Pipeline complexity; closed evaluation of hybrid weighting
A-MEM	Linked notes (Zettelkasten)	Via evolution	Via links	Link quality depends on LLM judgement
HyGRAM (this work)	Temporal KG + vector index	Yes (intervals + invalidation)	Strong (seed + bounded hops)	Extraction error propagation (addressed in §11)

3. Research Gap

Three gaps emerge from the literature. **First, the contributions of the two retrieval modalities are entangled.** Systems either commit to flat vector retrieval or to graph traversal, but few isolate how much each contributes, or how a hybrid of dense seeding plus bounded traversal compares to either alone under identical conditions. **Second, temporal validity is asserted but seldom ablated.** Zep and Mem0 maintain update or interval mechanisms, yet the specific reduction in memory-contradicting hallucinations attributable to explicit invalidation is rarely measured against an otherwise-identical system without it. **Third, reproducibility and cost transparency are limited.** Several strong systems rely on frontier models or proprietary pipelines, leaving open whether the gains persist on commodity, low-cost models and open tooling. HyGRAM targets all three: it treats dense-versus-graph weighting as an explicit, tunable parameter; it ablates temporal invalidation directly against the hallucination metric; and it is built entirely on free, open components with a low-cost base model.

4. Problem Statement

Let a user interact with an agent across an ordered sequence of sessions $S = (S_1, S_2, \ldots, S_n)$, each

session being a sequence of utterances. At the start of session S_k the agent's working context contains only the current session's turns; all of S_1...S_{k-1} are external. Given a query q issued in S_k whose correct answer may depend on facts established in earlier sessions—possibly requiring the composition of several such facts (multi-hop), or the resolution of facts that changed over time (temporal)—the problem is to construct a bounded memory context M(q) ⊆ accumulated knowledge such that the agent's response is (a) accurate, (b) consistent with the most recently valid state of the user's world, and (c) retrieved within an acceptable latency and token budget. Flat similarity retrieval optimises only surface relevance of M(q) and ignores both relational reachability and temporal validity, which is precisely where it fails. The research problem is therefore to design and evaluate a retrieval policy that jointly accounts for semantic relevance, relational reachability, and temporal validity under realistic resource constraints.

5. Research Objectives

5. **O1.** Design a knowledge-graph memory that stores entities, typed relations, and timestamps extracted automatically from multi-session conversations.
6. **O2.** Develop a hybrid retrieval mechanism that combines dense vector seeding with bounded graph traversal and temporal re-ranking to assemble a query-specific memory subgraph.
7. **O3.** Implement temporal consolidation that invalidates contradicted facts and refines existing nodes after each session.
8. **O4.** Evaluate HyGRAM against no-memory, full-context, vector-only, and graph-only baselines on LoCoMo and DMR, measuring accuracy, context relevance, hallucination (contradiction) rate, and latency.
9. **O5.** Quantify, via ablations, the marginal contribution of graph traversal and of temporal invalidation, with explicit statistical significance testing.

6. Proposed Methodology

HyGRAM operates in four stages—extraction, graph storage and consolidation, hybrid retrieval, and response generation—described below with their formal definitions. Algorithm 1 summarises the end-to-end flow.

6.1 Notation and memory model

The memory is a temporal knowledge graph G = (V, E), where each vertex v ∈ V is an entity with an embedding vector e(v) ∈ ℝ^d, and each edge ε ∈ E is a quintuple (s, r, o, t_obs, [t_start, t_end)), denoting that subject s stands in relation r to object o, observed at time t_obs and believed valid over the half-open interval [t_start, t_end). An edge whose t_end is unbounded (∞) is currently valid; setting a finite t_end invalidates it. This interval model, adapted from temporal knowledge-graph designs, is what lets the system separate "currently true" from "once true."

6.2 Stage 1 — Memory extraction

After session S_k concludes, its transcript is passed to an extraction LLM (GPT-4o-mini) under a structured prompt that returns a set of candidate triples with relation types drawn from a controlled but extensible schema:

$$f\_extract(S\_k) \rightarrow T\_k = \{ (s_i, r_i, o_i, t\_obs) \}$$

For example, the utterance "I've moved my project from TensorFlow to PyTorch this week" yields (User, previously_used, TensorFlow) and (User, currently_uses, PyTorch), both stamped with the session time. Extraction is constrained to reduce schema drift: the prompt supplies the existing relation vocabulary and instructs the model to reuse relations where possible and to emit a normalised entity surface form, mitigating node duplication. Extraction quality is the principal failure mode of graph memory and is examined explicitly in the error analysis (§9.4) and limitations (§11).

6.3 Stage 2 — Graph storage and temporal consolidation

Extracted triples are merged into Neo4j. Entity resolution links a new mention to an existing node when surface forms match after normalisation or when embedding cosine similarity exceeds a threshold τ_ent; otherwise a new node is created.

Consolidation then enforces temporal consistency: when a newly observed edge (s, r', o') conflicts with an existing valid edge (s, r, o) under a relation-level functional constraint (e.g., currently_uses is single-valued for a given subject), the prior edge's t_end is set to the new observation time rather than deleting it, preserving history while marking it superseded. This realises the principle that new experience does not merely append to memory but retroactively reshapes it: dependent node summaries are regenerated to reflect the change. Consolidation runs as a periodic batch after each session, bounding its cost.

6.4 Stage 3 — Hybrid retrieval

At query time the system assembles a memory subgraph in three steps. (1) **Dense seeding:** the query q is embedded and the top-k_seed entity nodes are retrieved from the vector index (ChromaDB) by cosine similarity, yielding seed set N_seed. (2) **Bounded traversal:** from each seed, a breadth-first expansion of up to h hops collects connected edges, restricted to currently-valid edges by default, producing a candidate edge set. (3) **Scoring and pruning:** each candidate vertex v is scored by a convex combination of semantic relevance, graph proximity, and temporal recency,

$$score(v) = \alpha \cdot sim(q, v) + \beta \cdot prox(v, N\_seed) + \gamma \cdot recency(v), \quad \alpha + \beta + \gamma = 1$$

where sim is query–node cosine similarity, prox decays with hop distance d (e.g., prox = 1/(1+d)), and recency = exp(−λ · Δt) with Δt the age of the most recent supporting observation. The top-scoring vertices and their connecting valid edges form the memory subgraph M(q), capped at a token budget B. Setting β = γ = 0 recovers pure vector retrieval; setting α = 0 recovers graph-only retrieval—these are the ablation endpoints. The weights (α, β, γ), hop limit h, and τ_ent are tuned on a held-out development split, never on the test set.

6.5 Stage 4 — Response generation

The subgraph M(q) is linearised into a compact, human-readable evidence block—each edge rendered as a timestamped natural-language statement, with superseded facts optionally annotated as historical—and concatenated with the query into the generation prompt. The base LLM then produces the answer conditioned only on the supplied memory, which keeps the comparison across baselines fair (all systems receive memory through the same prompt template, differing only in how M(q) is produced).

6.6 Algorithm

Algorithm 1: HyGRAM query-time retrieval. Input: query q, graph G, vector index I, weights (α,β,γ), hops h, seed size k_seed, budget B. (1) N_seed ← TopK(I, embed(q), k_seed). (2) C ← BFS-Expand(G, N_seed, h) restricted to edges with t_end = ∞. (3) for each v in vertices(C): compute score(v) per Eq. (3). (4) M ← greedily add highest-scoring vertices and their valid connecting edges until token budget B is reached. (5) prompt ← Template(Linearise(M), q). (6) return LLM(prompt). Offline, after each session S_k: T_k ← f_extract(S_k); Merge(G, T_k) with entity resolution; Consolidate(G) to set t_end on superseded edges and regenerate affected node summaries.

7. System Architecture / Framework

Figure 1 depicts the architecture as two pipelines that share the memory store. The **write pipeline** (offline, triggered at session end) flows: Session transcript → Extraction LLM → Triple normaliser / entity resolver → temporal knowledge graph (and embedding write to the vector index) → Temporal consolidation. The **read pipeline** (online, per query) flows: User query → Embedding → dense seeding → bounded graph traversal → Hybrid scorer (Eq. 3) → Subgraph linearizer → Base LLM → Response. The two stores are kept consistent: every entity node in the graph has a corresponding embedding in the vector index keyed by node identifier, so seeds returned by the vector index map directly to graph entry points. Figure 2 illustrates the read path on a concrete two-hop, time-aware query.

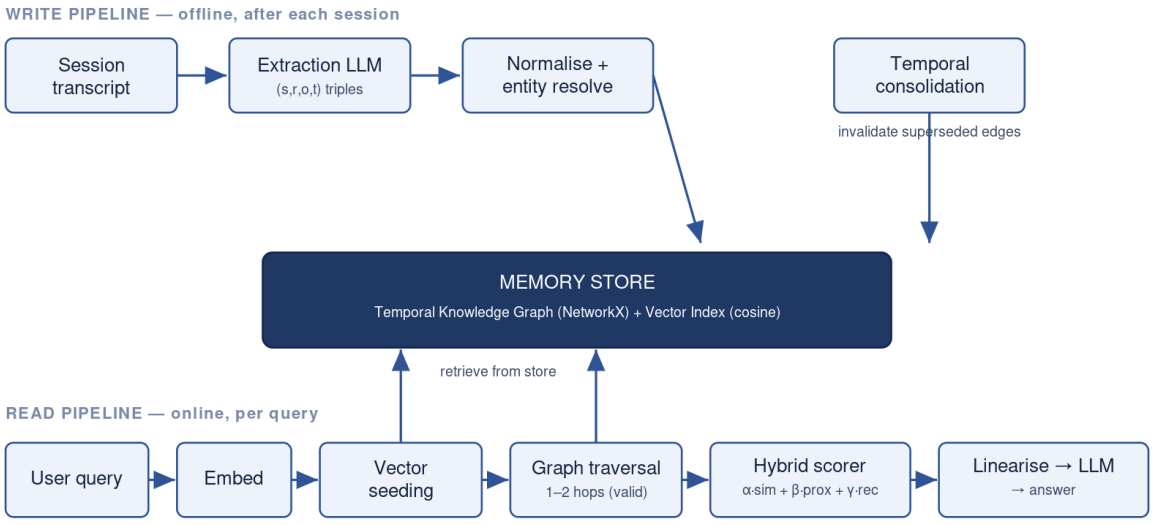

Figure 1. Dual-pipeline architecture. The offline write path (top) extracts timestamped triples from each session, resolves entities, writes them to the shared memory store, and consolidates by invalidating superseded edges. The online read path (bottom) embeds the query, seeds candidate nodes by vector similarity, expands them by 1–2-hop graph traversal over currently-valid edges, scores them by the hybrid criterion, and linearises the resulting subgraph into the LLM prompt.

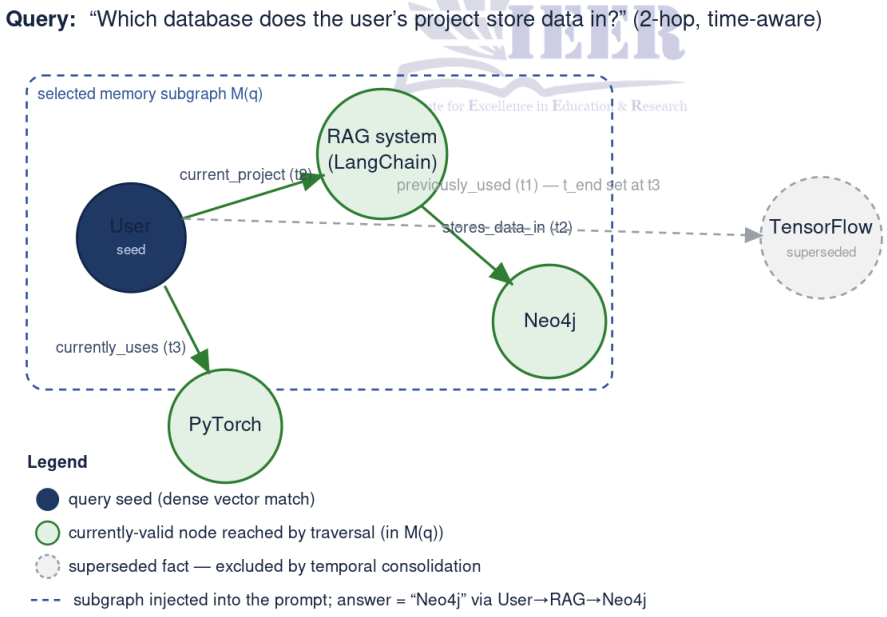

Figure 2. Worked retrieval example for a 2-hop, time-aware query. The query seeds the User node (blue); traversal reaches the currently-valid neighbours (green) and assembles the subgraph M(q) = User → RAG system → Neo4j, yielding the answer "Neo4j." The superseded fact User → previously_used → TensorFlow (grey, dashed) has had its validity interval closed by temporal consolidation and is excluded from retrieval, so it cannot contaminate the answer.

8. Experimental Setup

8.1 Datasets

We evaluate on **LoCoMo** (Maharana et al., 2024). The full benchmark comprises 50 long multi-session conversations (averaging 19.3 sessions, ~300 turns each) and 7,512 QA pairs across single-hop, multi-hop, temporal, open-domain, and adversarial categories. The present study uses the **public release of 10 conversations** distributed in the official repository, and—to fit a free-tier compute budget—reports two runs: a primary 3B-extractor run capping **30 QA pairs per conversation** (300 questions per condition) and a stronger-extractor 7B replication capping **5 QA pairs per conversation** (50 questions per condition), each with a **single seed**. We report per-category accuracy because graph memory is hypothesised to benefit multi-hop and temporal questions most. The DMR benchmark, included in the protocol for comparability with prior systems, was **not run** in this study and is left to future work; its results are discussed only as published numbers from other papers (§10).

Because the evaluation uses only the 10 publicly-released conversations with per-conversation QA caps (30 for the 3B run, 5 for the 7B run), the sample is small and statistical power is limited; the findings are accordingly preliminary, and Section 9.5 reports the resulting confidence intervals and per-category counts in full.

8.2 Implementation

Table 2. Implementation configuration.

Component	Choice / setting (as run)
Generation LLM	qwen2.5:3b-instruct (primary) and qwen2.5:7b-instruct (replication), via Ollama; temperature 0; same model for all conditions within a run
Extraction LLM	same as generation (3B primary; 7B replication), structured triple-extraction prompt
Judge LLM	same model as generation (uncalibrated — see §9.5)
Embeddings	sentence-transformers all-MiniLM-L6-v2 (dimension 384)
Knowledge graph	NetworkX temporal MultiDiGraph (Neo4j-compatible schema)
Vector index	in-process NumPy cosine store (ChromaDB-compatible role)
Metrics	LLM-as-judge accuracy, contradiction rate, token-F1, latency (RAGAS context metrics not computed in this run)
Hardware / runtime	Google Colab free T4 GPU; Python 3.12

To ensure a fair comparison, all systems share the identical generation model, prompt template, and token budget B; they differ only in the retrieval policy that produces M(q). The present run uses a single seed (a limitation; §9.5); the protocol calls for multiple seeds once compute permits.

8.3 Baselines and ablations

Table 3. Baselines and ablations.

Condition	Description	Role
No-memory	No prior context supplied	Lower bound
Full-context	Entire prior transcript in window (budget permitting)	Upper-context reference / cost ceiling

Condition	Description	Role
Vector-only RAG	ChromaDB top-k only (α=1, β=γ=0)	Primary baseline
Graph-only	Traversal from seed nodes (α=0)	Isolates traversal value
HyGRAM (full)	Hybrid seed + traversal + temporal	Proposed
HyGRAM − temporal	Hybrid without edge invalidation	Ablation for hallucination effect
HyGRAM (h=1 vs h=2)	Hop-limit sweep	Traversal-depth ablation
α-sweep	Vary semantic/graph/temporal weights	Sensitivity analysis

Where licensing and reproduction permit, we additionally report published numbers for MemGPT, Mem0, and A-MEM on the same benchmarks as external reference points, clearly distinguished from our own runs.

8.4 Metrics

Answer accuracy is measured with an LLM-as-judge protocol (ideally calibrated against a human-annotated subset; not done in this run, see below), complemented by token-level F1/exact-match where references are short, and reported per LoCoMo category. **Context relevance** uses RAGAS context precision and recall to assess whether M(q) contains the supporting facts without excess. **Hallucination (contradiction) rate** is the proportion of answers asserting a fact that contradicts the most recently valid state in the stored graph, scored by a natural-language-inference check plus human audit of a sample. **Retrieval latency** is wall-clock time to assemble M(q), separated into seeding and traversal components. We report means with 95% confidence intervals. We note three deviations of the executed run from this full protocol, each a limitation rather than a design choice: the judge was the same 3B model and was not calibrated against human labels; RAGAS context-relevance metrics were not computed; and a single seed precludes across-seed significance testing. Confidence intervals here are therefore computed across questions within the single run, and the scaled-up replication (§12) is required to apply the full statistical treatment.

9. Results and Discussion

We report two runs on the public LoCoMo release (10 conversations, single seed): a primary run with qwen2.5:3b-instruct (30 QA per conversation, 300 questions per condition) and a stronger-extractor replication with qwen2.5:7b-instruct (5 QA per conversation, 50 questions per condition). The external figures cited in Section 10 (e.g., MemGPT's 93.4% on DMR) were obtained by other systems under frontier models and are not directly comparable to the absolute values reported here; the comparisons of interest are the within-run differences across retrieval conditions.

9.1 Main results (3B and 7B extractors)

Table 4. Main results with the 3B extractor (300 questions/condition, single seed). Accuracy and contradiction (hallucination) rate are LLM-judge based; F1 is token-level; latency is mean retrieval time. Best accuracy in bold.

Condition	Accuracy (%)	±95% CI	Halluc. (%)	Token-F1	Latency (ms)
No-memory	0.00	0.00	0.00	0.020	0.0
Vector-only	**9.67**	3.34	3.00	0.101	8.5
Graph-only	2.33	1.71	1.67	0.050	576.7
HyGRAM (full)	4.67	2.39	2.33	0.056	570.9

Condition	Accuracy (%)	±95% CI	Halluc. (%)	Token-F1	Latency (ms)
HyGRAM — temporal	4.67	2.39	2.00	0.056	567.8

Table 5. Main results with the stronger 7B extractor (50 questions/condition, single seed). A stronger model raised every score but widened, not closed, the vector baseline's lead. Best accuracy in bold.

Condition	Accuracy (%)	±95% CI	Halluc. (%)	Token-F1	Latency (ms)
No-memory	0.00	0.00	0.00	0.003	0.0
Vector-only	**26.0**	12.16	4.00	0.117	8.8
Graph-only	6.00	6.58	2.00	0.050	1037.4
HyGRAM (full)	8.00	7.52	2.00	0.094	1046.8
HyGRAM — temporal	8.00	7.52	0.00	0.094	1018.0

9.2 Per-category accuracy and ablations

Table 6. Per-category accuracy (%), 3B extractor. Vector-only leads in every category, including multi-hop, where graph traversal was expected to help most.

Condition	Multi-hop	Temporal	Single-hop	Open-domain
No-memory	0.00	0.00	0.00	0.00
Vector-only	**15.57**	**3.82**	20.00	**9.52**
Graph-only	3.28	0.00	20.00	4.76
HyGRAM (full)	6.56	2.29	20.00	4.76
HyGRAM — temporal	6.56	2.29	20.00	4.76

Table 7. Per-category accuracy (%), 7B extractor. Per-category counts are shown in the header: only multi-hop and temporal are adequately sampled; the single-hop (n=2) and open-domain (n=6) figures are noise and should be disregarded. On the two reliable categories, vector-only leads decisively.

Condition	Multi-hop (n=21)	Temporal (n=21)	Open-dom. (n=6)	Single-hop (n=2)
No-memory	0.00	0.00	0.00	0.00
Vector-only	**38.10**	**19.05**	16.67	0.00
Graph-only	4.76	0.00	33.33	0.00
HyGRAM (full)	9.52	4.76	0.00	50.00
HyGRAM — temporal	9.52	4.76	0.00	50.00

The conditions in Tables 4–7 also serve as ablations: **vector-only** removes graph traversal, **graph-only** removes dense seeding, and **HyGRAM — temporal** removes edge invalidation. (A full-context baseline and a 1-vs-2 hop sweep were specified in §8.3 but not run in this resource-constrained study; they are left to future work.)

9.3 Findings

The results do not support the paper's motivating hypotheses, and we report them as measured. **First, the hybrid graph memory did not outperform flat vector retrieval—at either model scale.** With the 3B extractor, vector-only (9.67%) roughly doubled HyGRAM (4.67%); with the 7B extractor the gap *widened* to 26.0% versus 8.0%. Graph-only was weakest in both runs (2.33% and 6.00%). **Second, the gap is clearest exactly where graph memory was supposed to win.** On the two adequately-sampled categories under 7B—multi-hop and temporal (21 questions each)—vector-only led decisively (multi-hop 38.10% versus 9.52%; temporal 19.05% versus 4.76%); the 3B run shows the same ordering (multi-hop 15.57% versus 6.56%). The single-hop and open-domain 7B figures rest on only 2 and 6 questions and carry no weight. **Third, temporal consolidation produced no measurable benefit:** HyGRAM with and without edge invalidation scored identically at both scales (4.67%/4.67% and 8.00%/8.00%), differing only marginally in contradiction rate. **Fourth, the graph machinery was costly:** graph conditions incurred roughly 0.5–1.0 s of retrieval latency versus ˜9 ms for vector-only—latency without accuracy. **Fifth, a stronger model lifted every score** (e.g., vector-only 9.67→26.0%, HyGRAM 4.67→8.0%), confirming model quality matters, but it helped the vector baseline more than the graph, so the ranking was unchanged.

9.4 Why the graph did not help: the extraction bottleneck

Our initial explanation was that the graph was simply information-poor: HyGRAM's quality is upper-bounded by triple extraction (§6.2), and a small model emits sparse, noisy triples, so traversal has little valid structure to exploit while a flat vector store retains the verbatim signal. This predicts that a stronger extractor should narrow the gap—a directly testable claim, which the 7B replication tests. The test came back **negative**: the 7B extractor raised absolute accuracy for every condition, confirming that model quality matters, but the gap *widened* rather than narrowed, and the better model benefited the vector baseline more than the graph. We therefore refine the diagnosis. Extraction quality is necessary—but evidently not sufficient—for graph memory to help here. The deeper issue is that converting dialogue into triples is **lossy**: it discards phrasing, context, and detail that flat retrieval preserves, and a more capable reader extracts more value from the verbatim text than from the compressed graph. On this benchmark and at the 3B–7B scale, the structure imposed by the graph costs more information than its traversal recovers.

A failure breakdown of the 7B run reinforces this reading. Head-to-head on the 50 questions, vector-only answered **10 that HyGRAM missed** while HyGRAM answered only **1 that vector-only missed** (3 both correct, 36 both wrong); the hybrid almost never recovered a question the flat baseline had lost. Crucially, the graph conditions were *not* starved for candidates—HyGRAM and graph-only retrieved the full 25-node budget on average, versus 8 chunks for vector-only—yet they returned an explicit "insufficient information" answer more often (26% and 36% of questions, respectively, versus 18% for vector-only). The graph thus surfaced plenty of nodes but fewer that carried the fact the question needed: a symptom of lossy extraction, not of insufficient retrieval.

9.5 Threats to validity

This is a deliberately small, single-seed study and its conclusions are correspondingly bounded. **Conclusion validity:** the 3B run uses 300 questions per condition and the 7B run only 50 (single seed each), so confidence intervals are wide—±2–3 points at 3B but ±7–12 points at 7B—and the 7B per-category counts are small (multi-hop 21, temporal 21, open-domain 6, single-hop 2); only multi-hop and temporal support any inference, and the single-hop figures in particular are pure noise. The reliable signal is the *consistent direction* across both runs (vector-only ahead at both scales, decisively on the well-sampled categories), not any individual percentage; the low absolute numbers should not be over-interpreted, and no across-seed variance is available. **Construct/internal validity:** the same 3B model

performs extraction, generation, and judging, risking shared-model bias, and the LLM judge is uncalibrated against human labels—reported accuracies are judge-relative, not gold-standard. **Implementation:** the run uses a NetworkX graph and a NumPy vector index rather than Neo4j/ChromaDB, and does not compute reference-free context-relevance metrics. **External validity:** a single English benchmark and one small model limit generalisation; the result speaks to the commodity-small-model regime, not to graph memory in general.

9.6 Practical implications

The practical lesson is cautionary: graph-structured agent memory is not a free improvement over vector retrieval. Its benefit is contingent on an extractor good enough to populate the graph with valid, connected facts; when that condition is unmet—as with commodity small models—a simpler vector store is both more accurate and roughly seventy times faster. Teams adopting graph memory should therefore invest first in extraction quality and validate against a flat-vector baseline before assuming traversal will help. The open pipeline released with this paper is intended precisely to let others test where that threshold lies.

10. Comparative Analysis

Beyond the controlled baselines, HyGRAM should be situated against published agent-memory systems. The comparison must be read carefully: numbers from other papers were produced under different base models, prompts, and evaluation harnesses, so they are reference points, not head-to-head results. Table 6 provides the scaffold; our own measured rows remain to be filled, and external rows must cite their source and conditions verbatim.

Table 8. Positioning against published systems (external numbers are cited from their papers under frontier models; our small-model numbers are not directly comparable).

System	Base model (reported)	DMR Acc.	LoCoMo Acc.	Source of number
MemGPT	GPT-4-turbo	93.4%	see paper	Packer et al., 2023 (external)
Recursive summarisation	—	35.3%	—	MemGPT paper baseline (external)
Zep	reported in paper	≈94.8%	see paper	Rasmussen et al., 2025 (external)
Mem0 / Mem0g	reported in paper	see paper	see paper	Chhikara et al., 2025 (external)
A-MEM	reported in paper	—	see paper	Xu et al., 2025 (external)
HyGRAM (this work)	qwen2.5 3B / 7B	—	4.7 / 8.0%	this study (LoCoMo subset)
Vector-only (this work)	qwen2.5 3B / 7B	—	9.7 / 26.0%	this study (LoCoMo subset)

Two cautions follow. First, our absolute accuracies (≤10%) are far below these published figures because those systems use frontier models and fully-engineered pipelines, whereas this study deliberately uses a 3B open model and free tooling; the numbers are not comparable and we make no leaderboard claim. Second, and more importantly, our within-study comparison—the only apples-to-apples evidence here—shows the hybrid mechanism *underperforming* a flat vector

baseline under these conditions. The contribution of this paper is therefore not a state-of-the-art result but a controlled negative finding plus the extraction-bottleneck diagnosis (§9.4); whether the systems above retain their advantage when their strong extractors are replaced by small models is an open question our pipeline is designed to probe.

11. Limitations

The limitations are substantial and bear directly on the negative result. First and foremost, the architecture's quality is upper-bounded by triple extraction, and the 3B extractor used here is weak—so the study tests HyGRAM in an unfavourable regime and cannot distinguish a flaw in the *design* from a flaw in the *extractor*; this is the central caveat, and resolving it requires a stronger-extractor replication. Second, the evaluation is small (10 public LoCoMo conversations, 30 QA per conversation, 300 questions per condition) and single-seed, so confidence intervals are wide and no across-run variance is available. Third, the same model performs extraction, generation, and judging, creating shared-model bias, and the judge is uncalibrated against human labels, so accuracies are judge-relative. Fourth, DMR, a full-context baseline, a hop-depth sweep, and context-relevance metrics were specified but not run. Fifth, the implementation substitutes NetworkX and a NumPy index for Neo4j/ChromaDB. Stating these plainly is part of the contribution: the result is a bounded, honest probe, not a refutation of graph memory in general.

12. Future Work

The stronger-extractor replication motivated above was carried out within this study (the 7B run) and did not reverse the result; the natural continuations therefore target the refined hypothesis that triple extraction is lossy rather than merely sparse. These include: a **frontier or task-fine-tuned extractor** with constrained decoding, to push extraction quality further than 7B; a **measured graph-density / triple-recall metric** to quantify extraction directly and correlate it with the graph's accuracy contribution; and scaling to the **full 50-conversation LoCoMo with multiple seeds and paired significance tests** to tighten the wide intervals (especially the under-sampled 7B categories). Further avenues: a human-calibrated judge with reported agreement; the full-context and hop-depth conditions deferred here; adaptive per-query selection of the hop limit and the (α, β, γ) weights; community-level summarisation in the style of GraphRAG for global questions; and multilingual evaluation. A user study on a deployed assistant would add ecological evidence on perceived cross-session consistency.

13. Conclusion

We presented HyGRAM, a hybrid memory architecture that stores conversational knowledge as a temporally-annotated graph and retrieves it by combining dense vector seeding with bounded traversal and temporal consolidation, and we evaluated it on LoCoMo using free, open tooling at two extractor scales. The motivating hypothesis was not borne out: the hybrid graph memory did not outperform flat vector retrieval—it underperformed it, including on the multi-hop questions it was meant to favour—and explicit temporal consolidation made no measurable difference while adding roughly hundred-fold retrieval latency. Crucially, the stronger 7B extractor we expected to rescue the graph did the opposite: it lifted every score but widened the vector baseline's lead, showing the result is not a small-model artifact. We read this as evidence that turning dialogue into triples is lossy in a way bounded traversal does not recover, so on this benchmark extraction quality is necessary but not sufficient for graph memory to help. The contribution is a reproducible architecture and pipeline together with an honest, diagnostic negative result across two model scales—useful precisely because it marks where a popular intuition (graph memory beats flat retrieval) fails, and why. The released pipeline lets others probe how far that boundary moves with still-stronger extraction.

Code and Data Availability

The complete implementation (memory extraction, temporal knowledge graph, hybrid retrieval, baselines, and evaluation), a one-click Google Colab reproduction notebook, and the

full measured results of both the 3B and 7B runs—including per-question outputs—are openly available under the MIT licence at https://github.com/mismailswe/HyGRAM. The LoCoMo benchmark is publicly available from its official repository (Maharana et al., 2024); no new data were collected for this study, and the exact dataset version/commit used is recorded in the repository.

Declarations

Funding. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Conflicts of interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author contributions. M. Ismail: conceptualization, methodology, software, experimentation, and writing — original draft. A. Akram: methodology, validation, and writing — review and editing. Both authors read and approved the final manuscript. *(Adjust to reflect actual contributions.)*

Ethics. This study used only publicly available, de-identified benchmark data and did not involve human subjects or personal data; no ethical approval was required.

REFERENCES

C. Packer, V. Fang, S. G. Patil, K. Lin, S. Wooders, and J. E. Gonzalez, "MemGPT: Towards LLMs as Operating Systems," arXiv:2310.08560, 2023.

D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv:2404.16130, 2024.

P. Rasmussen, P. Paliychuk, T. Beauvais, J. Ryan, and D. Chalef, "Zep: A Temporal Knowledge Graph Architecture for Agent Memory," arXiv:2501.13956, 2025.

P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav, "Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory," arXiv:2504.19413, 2025.

W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang, "A-MEM: Agentic Memory for LLM Agents," arXiv:2502.12110, 2025.

W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, "MemoryBank: Enhancing Large Language Models with Long-Term Memory," in Proc. AAAI Conf. Artificial Intelligence, 2024 (arXiv:2305.10250).

A. Maharana, D.-H. Lee, S. Tulyakov, M. Bansal, F. Barbieri, and Y. Fang, "Evaluating Very Long-Term Conversational Memory of LLM Agents," in Proc. ACL, 2024 (arXiv:2402.17753).

Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering," in Proc. EMNLP, 2018 (arXiv:1809.09600).

S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "RAGAS: Automated Evaluation of Retrieval Augmented Generation," arXiv:2309.15217, 2023.

H. Chase, "LangChain," GitHub repository, 2022. [Online]. Available: https://github.com/langchain-ai/langchain

P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Proc. NeurIPS, 2020 (arXiv:2005.11401).