Reverie: A Two-Layer Architecture for Persistent AI Memory and the Saturation of LongMemEval

Waleed Abdullah
Independent Researcher

March 2026

Abstract

Reverie achieves 94.6% on LongMemEval (n=500, GPT-4o judge), within 0.27 points of the top-performing system. A controlled Oracle experiment (running the same synthesis model, Claude Sonnet 4.6, with perfect retrieval) scores 93.4%, revealing that LongMemEval is model-dominated: the architecture contributes +1.2 points, concentrated in knowledge-update and multi-session categories where architectural features (supersession tracking, session summaries) directly apply. This pattern is not unique to Reverie; we estimate comparable architectural deltas across leaderboard systems. The system is a two-layer memory architecture: L1 stores raw conversational experiences losslessly, and L2 extracts declarative facts with LLM-confirmed supersession detection for knowledge updates. Both layers are searched with hybrid vector+keyword retrieval and synthesized by an LLM. The paper's primary contribution is methodological: an iterative build-test-prune development process in which every component was subjected to ablation, and several (including three additional layers, a prediction layer, weight decay, contextual embeddings, and LLM-declared edges) were removed when they degraded performance or failed to justify their complexity.

1 Introduction

Every AI conversation starts from zero. Users re-explain their context, preferences, and history each session. An assistant that spent hours helping plan a kitchen renovation forgets the layout, the budget, and the user's material preferences the moment the session ends. This is not merely inconvenient; it is fundamentally limiting. As interaction histories grow to thousands of sessions, they exceed even the largest context windows, making raw history injection impossible regardless of cost.

Current approaches to persistent memory (context-window injection [Liu et al., 2023], flat RAG, observation logging, and knowledge graphs) each face significant limitations in temporal reasoning, knowledge update tracking, or lossless retention (Section 2).

We present Reverie, a two-layer memory architecture. L1 stores raw conversational experiences losslessly. L2 extracts declarative facts using an LLM, with supersession detection to track knowledge updates. Both layers are searched with hybrid vector+keyword retrieval and synthesized by an LLM. The architecture is governed by design principles that emerged from extensive empirical testing, including building and removing four additional layers that failed to improve performance (Section 3.1, Appendix A).

Our contributions are:

- An **empirical build-test-prune methodology** for memory architecture development, documented through comprehensive ablation studies. Several components (including four ad-

ditional layers, weight decay, contextual embeddings, and LLM-declared edges) were built, tested, and removed when empirical evaluation showed them to be ineffective (Appendix A). The architecture and benchmark result are outputs of this process.

- A **two-layer architecture** that achieves 94.6% on LongMemEval using only experience storage and fact extraction with supersession tracking.
- **LLM-confirmed supersession detection**, a two-stage pipeline (vector candidate selection + LLM confirmation) for tracking knowledge updates that handles the heavily overlapping similarity distributions between genuine updates and unrelated similar facts.
- A **model-dominance finding**: Oracle experiments reveal that LongMemEval performance is primarily determined by synthesis model quality, with architectures contributing single-digit improvements over a strong baseline.

2 Related Work

Long-term memory for conversational AI has been approached from several directions, which we organize by their core storage and retrieval strategies.

Extracted memory facts. Mem0 [Chhikara et al., 2025] extracts and consolidates "memory facts" from conversations, achieving 66.9% on the LoCoMo benchmark [Maharana et al., 2024]. Mem0's fact extraction and deduplication pipeline is well-engineered and mirrors our L2 fact layer. However, Mem0 discards the raw conversation after extraction, a lossy step that prevents recovery when the extraction model misses relevant information. Reverie retains raw experiences at L1 and treats extracted facts as an enrichment layer.

Temporal knowledge graphs. Zep/Graphiti [Rasmussen et al., 2025] builds a temporal knowledge graph with LLM-declared entity relationships, achieving 71.2% on LongMemEval. The approach provides valuable explicit temporal metadata and entity tracking, and its graph structure enables relationship-aware queries. However, it relies on write-time LLM edge declaration, which we found to produce unreliable graph structures in our own experiments (Appendix A.2).

Memory-as-operating-system. MemGPT/Letta [Packer et al., 2023] treats memory as an operating system with explicit read/write operations managed by the agent itself. This provides fine-grained control and a principled abstraction for memory management. However, it adds significant complexity (the agent must decide when and what to store) and lacks the temporal metadata and structured retrieval mechanisms needed for knowledge-update and temporal-reasoning queries.

Observational memory. Mastra Observational Memory [Barnes, 2025] deploys observer and reflector agents that watch conversations and maintain structured observation logs. It is the current leader on LongMemEval, scoring 84.23% with GPT-4o and 94.87% with GPT-5-mini. The observer/reflector design is elegant and produces compact, semantically rich memory representations. However, the approach is lossy by design: raw conversation is replaced by observations, and anything the observer misses is unrecoverable. All memory systems exhibit some degree of model sensitivity; the observation approach is particularly exposed because both write-time observation quality and read-time synthesis depend on the underlying model.

Retrieval-optimized approaches. Emergence AI [Emergence AI, 2025] achieves 86% on LongMemEval with GPT-4o using a RAG architecture whose key insight (retrieving entire sessions rather than individual turns, scored by NDCG) is effective and simple. Supermemory [Supermemory, 2025] uses atomic memory extraction with source chunk injection, reaching 85.2% with Gemini 3 Pro. Hindsight [Vectorize, 2025] combines biomimetic memory with structured entity and temporal awareness, achieving 91.4% with Gemini 3 Pro, the strongest result among non-observation approaches prior to Reverie. These systems demonstrate strong retrieval engineering but do not

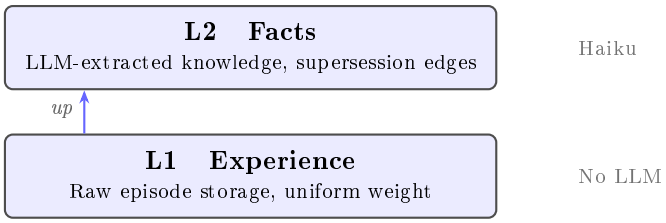

Figure 1: Reverie's two-layer architecture. L1 stores raw conversational turns losslessly. L2 extracts declarative facts via Haiku and tracks knowledge updates through supersession edges. Upward signal flow carries extracted facts from L1 experiences. Both layers are searched with hybrid vector+keyword retrieval.

build persistent memory structures that evolve over time.

Benchmarks. LongMemEval [Wu et al., 2024] has emerged as the standard evaluation for conversational memory systems, testing six categories across 500 questions with synthetic conversation histories. LoCoMo [Maharana et al., 2024] provides complementary evaluation focused on very long conversations. More recently, MemoryAgentBench [Hu et al., 2025] introduced incremental multi-turn evaluation, which better tests temporal and knowledge-update capabilities but has seen limited adoption so far.

Gap addressed by Reverie. No existing system combines (1) lossless raw storage with fact extraction enrichment, (2) explicit LLM-confirmed supersession detection for knowledge updates, and (3) dual retrieval paths that scale from context-window recall for small archives to layered retrieval for archives exceeding context limits. Reverie achieves state-of-the-art-class results while maintaining non-lossy storage and explicit knowledge update tracking, properties that become increasingly valuable as archive scale grows beyond what current benchmarks test.

3 Architecture

Reverie is organized as a two-layer architecture. Information flows upward from raw experiences to extracted facts. The architecture separates a fast ingestion path (sub-millisecond `perceive`) from an offline consolidation pipeline where all expensive computation occurs.

3.1 Overview

The design follows four core principles that emerged from empirical testing:

1. **Store everything, abstract on top.** Raw experiences are never discarded or decayed. Ablation showed that weight decay was net destructive at all tested scales, permanently deleting information that may be relevant to rare future queries. All abstraction operates as additional layers over lossless storage: the system stores every stimulus without lossy compression, though completeness depends on the integration layer delivering all relevant conversational turns.
2. **Earn complexity, don't assume it.** Every component must justify its presence through measurable improvement. Four additional layers (association, abstraction, identity, prediction), contextual embedding enrichment, and LLM-declared edges were all built, tested, and removed when they failed to improve performance (Appendix A). The resulting two-layer architecture is the minimal system that produces the reported result.

3. **Fast ingestion, expensive consolidation.** The `perceive` operation stores raw text with no embedding computation (∼3 ms). All LLM calls, embedding, fact extraction, and supersession detection occur during offline `consolidate`, which can run asynchronously between sessions.
4. **Dual retrieval paths.** Small archives (< 400K characters, approximately 100K tokens, chosen to leave headroom in the synthesis model's 200K-token context window for prompt instructions and output) use context-window recall, sending all experiences chronologically to the synthesis model. Large archives use layered retrieval: hybrid vector+keyword search with fragment-based synthesis. The system automatically selects the appropriate path.

3.2 Data Model

The system operates on three core types:

Nodes represent information at either layer. Each node carries an embedding vector (384-dimensional, from all-MiniLM-L6-v2), a weight (0.0–5.0), a monotonic creation tick, and layer-specific metadata. L1 nodes store raw conversational turns. L2 nodes store extracted facts.

Edges encode relationships between nodes. Two edge types capture the architecture's structure: `extracted_from` (L2→L1, provenance tracking) and `supersedes` (L2→L2, knowledge updates where a new fact replaces an old one).

ReconstructedMemory is the output of the recall pipeline: an LLM-synthesized answer with source node provenance and confidence score.

3.3 Layer Details

Layer 1: Experience. Raw episode storage. Stores conversational turns verbatim with uniform weight. During consolidation, batch-embeds all unembedded nodes and generates session summaries for sessions with 3+ turns via Haiku, stored as L2 fact nodes. These summaries are critical for cross-session enumeration queries.

Layer 2: Facts. LLM-extracted declarative knowledge. Fact extraction is batched and deferred entirely to consolidation (no LLM calls during ingestion). L2 also performs supersession detection (Section 3.5).

3.4 Signal Flow

Perceive (fast path, ∼3 ms): L1 stores the raw experience with `embedding=None`. No LLM calls, no embedding, no computation.

Consolidate (offline, ∼25–45 s per session): The heavy processing pipeline. L1 batch-embeds all unembedded nodes and generates session summaries. L2 extracts facts in batches (parallel Haiku calls), builds embeddings, and runs supersession detection.

Recall: Routes queries through one of two paths based on archive size. For small archives (<400K chars), context-window recall sends all L1 experiences chronologically to Sonnet with session boundary markers and supersession metadata. For large archives, layered retrieval uses hybrid vector+keyword search across L1 and L2, supersession chain following, keyword re-ranking, and fragment-based LLM synthesis. On LongMemEval, all archives are small enough for context-window recall. We report results using `force_cortex=True` (the layered retrieval path) rather than context-window recall to demonstrate the architecture's viability at scales beyond context window limits, the regime where architectural choices become decisive.

3.5 Supersession Detection

When new information updates old information ("I have 3 cats" → "I have 4 cats"), the system must detect and track these knowledge updates. During L2 consolidation:

1. **Candidate selection:** For each new fact, vector search finds similar existing facts (top-5, cosine ≥ 0.65; threshold selected via manual inspection during development, not tuned on the evaluation set).
2. **LLM confirmation:** Parallel Haiku calls confirm whether the new fact genuinely updates the old one (binary yes/no).
3. **Edge creation:** Confirmed supersessions create a `SUPERSEDES` edge from new to old, and the old fact's weight is halved.

Confirmation calls are parallelized and use Haiku (the cheapest available model), so the $O(N \times K)$ cost remains tractable: a typical session yields 15–25 new facts × 5 candidates = 75–125 parallel Haiku calls, completing in under 5 seconds.

Empirical analysis of the similarity distributions (n=500 questions) showed that confirmed supersessions and rejected candidates have heavily overlapping similarity ranges (confirmed: 0.647–0.981, mean 0.734; rejected: 0.600–1.000, mean 0.710). Rejected candidates at similarity 1.0 represent identical facts mentioned across sessions without value changes (the same information restated, not updated). This overlap means pure threshold-based detection is insufficient; the LLM confirmation step is necessary and cannot be replaced by a higher similarity cutoff.

During retrieval, supersession metadata is used in two ways: (a) if a superseded fact appears in retrieved fragments but its replacement does not, the replacement is injected; (b) superseded facts are explicitly labeled as "OUTDATED" in the synthesis prompt so the LLM knows to prefer newer values.

3.6 Hybrid Search

Retrieval combines vector similarity and keyword matching:

$$\text{score} = 0.5 \times \text{cosine\_similarity}(q, n) + 0.5 \times \text{keyword\_fraction}(q, n)$$

Keyword matching uses simple suffix-stripping stemming (no external NLP dependencies) and stopword filtering. This hybrid approach ensures that queries containing proper nouns ("Ahmed") or specific terms ("Tiger I tank") are not lost to vector similarity averaging, while semantic queries ("how do they feel about competition") still benefit from dense retrieval.

Why plain embeddings suffice. A natural question is why plain text embeddings work when "chess" and "Ahmed" are semantically distant in embedding space. The answer is the interaction between hybrid search and L1 raw logs. L1 stores verbatim turns such as "Ahmed and I played chess at the cafe downtown," where both terms co-occur. Keyword search on L1 retrieves this turn for any query mentioning either "Ahmed" or "chess." L1 thus acts as contextual glue for entities that co-occur in verbatim text, providing the bridging that contextual embedding enrichment (Appendix A.3) attempted to inject artificially into L2 embeddings. This co-occurrence bridging does not solve taxonomic vocabulary gaps, where a query uses a category term ("model kits") but the text contains only a specific instance ("1/16 scale German Tiger I") with no shared keywords. That class of failure remains the primary unsolved retrieval challenge (Section 7).

For exhaustive/counting queries (detected by heuristic classification), an additional keyword scan with no top-K limit retrieves all potentially relevant nodes, and Haiku generates query expansion synonyms for a second retrieval pass.

4 Experimental Setup

4.1 Benchmark

We evaluate on LongMemEval [Wu et al., 2024], a benchmark of 500 questions testing six categories of long-term memory capability: single-session extraction (user, assistant, and preference), multi-session reasoning, knowledge updates, and temporal reasoning. Each question is paired with a synthetic conversation history of approximately 30–40 sessions (∼115K tokens). LongMemEval is the most widely-used benchmark for conversational agent memory and has been adopted by Mastra, Emergence AI, Zep, Supermemory, and others as the standard evaluation.

4.2 Configuration

- **Embedding model:** all-MiniLM-L6-v2 [Reimers & Gurevych, 2019]; 384 dimensions, off-the-shelf, no fine-tuning
- **Extraction model:** Claude Haiku 4.5 (fact extraction, supersession detection, session summaries)
- **Synthesis model:** Claude Sonnet 4.6 (memory synthesis)
- **Storage:** SQLite with WAL mode
- **Retrieval:** Hybrid vector + keyword search (50/50 weighting)

No models were fine-tuned. The system uses only off-the-shelf components: a public sentence-transformer for embeddings and the Anthropic API for LLM operations.

Reproducibility details. LLM temperature: not explicitly set, defaulting to Anthropic API provider behavior (1.0 at time of evaluation). Max tokens: 4096 for synthesis (recall), 1024 for extraction (Haiku tasks). Embedding: batch-embedded during consolidation (all unembedded nodes per layer). Supersession threshold: cosine similarity ≥ 0.65 (Section 3.5). Context-window budget: archives < 400K characters use context-window recall; larger archives use layered retrieval. Code is available upon request for research purposes. Evaluation logs (question-level predictions and judge decisions) are publicly available to enable independent verification of reported results.

4.3 Evaluation Protocol

Each question is processed independently with a fresh Reverie instance (no shared state across questions). The conversation history is ingested turn-by-turn via `perceive`, consolidated session-by-session, and then the question is answered via `recall` with `force_cortex=True` to ensure the layered retrieval path is exercised regardless of archive size.

Judge: GPT-4o with per-category binary yes/no prompts, matching the standard LongMemEval evaluation methodology used by all other systems on the leaderboard.

Query date injection: The benchmark's `question_date` field is prepended to each query as `[Today is {date}]`, providing the temporal reference point needed for questions involving relative time ("how many weeks ago").

5 Results

5.1 Main Results

Evaluated using the official LongMemEval GPT-4o judge with per-category prompts (identical to the evaluation used by all other systems on the leaderboard):

Category	n	Score
Single-session-assistant	56	**100%** (56/56)
Single-session-preference	30	**97%** (29/30)
Single-session-user	70	**97%** (68/70)
Knowledge-update	78	**97%** (76/78)
Temporal-reasoning	133	**92%** (123/133)
Multi-session	133	**91%** (121/133)
Overall	**500**	**94.6%** (473/500)

Table 1: Reverie results on LongMemEval (n=500). 95% Wilson confidence interval: [92.3%, 96.3%]. The difference between Reverie (94.6%) and Mastra OM (94.87%) is not statistically significant ($p > 0.05$, two-proportion z-test); the systems are statistically tied at this sample size.

5.2 Comparative Results

Rank	System	Model	Score
1	Mastra OM	GPT-5-mini	94.87%
2	**Reverie**	**Claude Sonnet 4.6**	**94.6%**
3	Mastra OM	Gemini 3 Pro	93.27%
4	Hindsight	Gemini 3 Pro	91.40%
5	Mastra OM	Gemini 3 Flash	89.20%
6	Hindsight	GPT-OSS 120B	89.00%
7	Emergence AI	GPT-4o	86.00%
8	Supermemory	Gemini 3 Pro	85.20%
9	Supermemory	GPT-5	84.60%
10	Mastra OM	GPT-4o	84.23%
11	Zep	GPT-4o	71.20%

Table 2: LongMemEval leaderboard. All scores use the standard GPT-4o judge. Rankings reflect combined architecture + synthesis model quality; a direct architecture comparison would require controlling for the synthesis model, which none of these evaluations do.

Reverie closes to within 0.27 points of the leader. We do not have Oracle data for GPT-5-mini, so we cannot directly compare architectural contributions between Reverie and Mastra OM.

Isolating architecture from model. The Oracle baseline provides the model with *only* the relevant sessions (perfect retrieval, no noise). To decompose Reverie's performance, we ran Oracle with the same synthesis model (Claude Sonnet 4.6):

Configuration	Score
Oracle + GPT-4o (leaderboard baseline)	82.4%
Oracle + Claude Sonnet 4.6	93.4%
Reverie (layered retrieval + Sonnet 4.6)	94.6%

Table 3: Oracle decomposition. The +12.2 point gap over the GPT-4o Oracle decomposes as +11.0 from model quality and +1.2 from architecture.

The 95% confidence interval on the architectural delta is $[-1.7\%, +4.1\%]$ (two-proportion z-test,

$p = 0.42$); the difference is not statistically significant at n=500. While the per-category sample sizes (n=78 to 133) are too small for independent significance testing, we observe directional alignment in categories where architectural features directly apply: knowledge-update (97% vs. 95% Oracle, from supersession tracking) and multi-session (91% vs. 88% Oracle, from session summaries). The Oracle baseline receives *only* the gold-relevant sessions (typically 1–3 of ∼40), while Reverie retrieves from the full archive; that it matches Oracle-level accuracy suggests architectural features (supersession metadata, session summaries) compensate for retrieval imperfection.

This pattern is not unique to Reverie. Mastra OM with GPT-4o scores 84.23%, just +1.8 over the GPT-4o Oracle baseline of 82.4% (reported by Wu et al., 2024 as the upper-bound reference in the LongMemEval evaluation framework). **LongMemEval appears to be model-dominated at current synthesis model capabilities**: architectures contribute single-digit improvements over a strong baseline, with the bulk of performance determined by synthesis model quality. This finding motivates evaluation at scales where context-window approaches are infeasible and architectural differences become decisive (Section 7).

5.3 Score Evolution

Version	Key Changes	Score
v1 (baseline)	Layered retrieval path, hybrid search	76%
v2	+ Session summaries, supersession chains, wider top-K	86%
v3	Switched to chronological synthesis (regressed; see Appendix A.3)	84%
v4	+ Supersession direction fix, preference prompt	88%
v5	+ Query date injection, temporal reasoning prompt	94.0%
v6 (final)	+ Synthesis precision prompts, bug fixes	**94.6%**

Table 4: Score evolution. v1–v4 used an internal keyword-matching evaluator; scores are not directly comparable to v5–v6 GPT-4o judge scores. We include them to illustrate the iterative methodology: each version's delta reflects a specific architectural change, even though absolute numbers shifted when switching to the GPT-4o judge.

The single largest improvement was query date injection (prepending `[Today is {date}]` to queries), which moved temporal reasoning from 74% to 90%, contributing approximately 6 points to the overall score. This finding was independently observed by Mastra, who reported a similar improvement when fixing their timestamp handling.

5.4 Ablation Studies

Note: The ablations below were conducted across different code versions and benchmark configurations, as each ablation was run at the point in development where it was most informative. Full n=500 ablations were prohibitive (∼$50/run), so most use smaller samples. We report the version and sample size for each.

Layered Retrieval vs. Context-Window. On LongMemEval n=50, the context-window path (all ∼40 sessions including noise) scored 86%. The layered retrieval path scored 92%, a +6 point improvement. The layered path outperforms context-window by filtering noise through retrieval and adding supersession labels and session summaries.

Category-level comparison (n=50, v6):

Category	Context-Window	Layered (v6)	Delta
Temporal-reasoning	100% (12/12)	100% (12/12)	0%
Knowledge-update	83% (10/12)	100% (12/12)	+17%
Single-session-pref	92% (12/13)	92% (12/13)	0%
Multi-session	69% (9/13)	77% (10/13)	+8%

Table 5: Category-level layered retrieval vs. context-window comparison. n=50 balanced sample covering the four categories where architectural features are expected to matter; single-session-assistant and single-session-user are omitted as both paths achieve near-identical scores on these categories. Category-level margins should be interpreted directionally given the small per-category counts (12–13). The context-window baseline was run at v5; the layered column is v6. Both use the same Sonnet version and GPT-4o judge.

Layered retrieval matches context-window on temporal reasoning and preference queries, and outperforms it on knowledge-update (+17%) and multi-session (+8%). The knowledge-update advantage reflects supersession tracking: the layered path explicitly labels outdated facts, while context-window relies on the LLM to notice conflicting information spread across sessions. The multi-session advantage reflects session summaries providing a compact enumeration index.

Decay ablation. Enabling multiplicative weight decay (even at low rates) was consistently net destructive across all benchmarks. Decay permanently removes information that may be relevant to rare queries. All node weight decay rates are set to 0.0.

6 Analysis

Of the 27 failures (using the standard GPT-4o judge), analysis identifies the following primary failure categories:

Temporal reasoning (10 failures, 7.5% of TR questions). The most common failure mode. Root causes split between retrieval and synthesis: fragment gaps where not all temporally relevant events are surfaced (e.g., "how many charity events before Run for Cure" retrieves 1 of 4), semantic confusion between temporal descriptions ("booked 3 months in advance" misinterpreted as "3 months ago"), and LLM date arithmetic errors where the model miscalculates durations despite having correct date labels. The arithmetic failures are a synthesis model limitation; a stronger model would likely resolve roughly half of them, but fragment gap cases require retrieval improvements.

Multi-session counting (12 failures, 9% of MS questions). Items described with different vocabulary from the query ("diorama featuring a 1/16 scale German Tiger I" vs. "model kits") resist both vector similarity and keyword matching. The synthesis LLM also occasionally hallucinates additional items or misses items present in the fragments.

Remaining failures (5). Knowledge update (2), single-session user (2), and preference (1). Root causes include retrieval failures where the relevant fragment was not surfaced and edge cases in temporal reference resolution.

Judge false negatives. Manual review of failures identified 3–5 probable false negatives where Reverie's answer is substantively correct but the GPT-4o judge marks it incorrect due to framing differences. For example, answering "0 times; no egg tarts are mentioned in the memories" to a question about baking egg tarts is logically equivalent to "not enough information" but is judged as incorrect. We report the official judge score without adjustment.

Synthesis ceiling. An estimated 60–70% of the remaining 5.4% failure rate reflects synthesis reasoning errors (date arithmetic, multi-hop counting) rather than retrieval gaps. We tested several

retrieval augmentations (session expansion, temporal range retrieval, pre-computed date arithmetic, entity graph traversal for counting queries) that uniformly degraded performance, consistent with context pollution: adding marginally-relevant content hurt synthesis precision without helping reasoning.

7 Limitations and Future Work

Forgetting mechanism. Node weight decay is disabled (all rates 0.0). At 10K+ nodes, retrieval noise from volume will become a concern. We plan to explore retrieval-time deprioritization (recency-weighted scoring) rather than storage-time destruction (weight decay), consistent with the "store everything" principle.

Vector search scaling. The current implementation loads all embeddings for a layer into a NumPy matrix for each query. At 100K nodes this becomes a bottleneck (∼150 MB per search). Migration to an approximate nearest-neighbor index (HNSW or IVF) is planned.

Consolidation speed. Consolidation runs synchronously. Batching supersession checks and parallelizing independent layer operations would significantly reduce this cost.

Single-user architecture. The current implementation uses single-file SQLite, limiting deployment to single-user, single-machine scenarios. Migration to PostgreSQL for multi-user support is straightforward given the relational schema.

Single-benchmark evaluation. All results are reported on LongMemEval. While the architecture's principles are general, specific prompts (temporal reasoning instructions, synthesis precision directives) and thresholds (0.65 supersession similarity) were refined on LongMemEval distributions. Cross-benchmark validation on LoCoMo [Maharana et al., 2024] or MemoryAgentBench [Hu et al., 2025] is needed to establish generalizability.

Scale evaluation. LongMemEval archives (∼115K tokens) fit within modern context windows, meaning context-window approaches are always viable. At these scales, architecture contributes minimally (Section 5.2). Evaluation at scales where architecture genuinely matters (hundreds of sessions, thousands of queries, archives exceeding context limits) is the key open question. We are developing a benchmark for this purpose.

Vocabulary bridging. Multi-session counting failures where items use different vocabulary from the query ("diorama featuring a 1/16 scale German Tiger I" vs. "model kits") are the primary unsolved retrieval challenge. Query expansion via synonym generation partially addresses this, but systematic vocabulary bridging likely requires graph-based approaches (entity co-reference, association networks) that we have not yet validated at scale.

8 Conclusion

This paper makes two contributions: an architecture and a finding about the benchmark used to evaluate it.

The architecture. Reverie achieves 94.6% on LongMemEval, within 0.27 points of the top-performing system. The design is empirically grounded: components that improved benchmarks were kept; components that failed to improve them (including four additional layers, weight decay, contextual embeddings, and LLM-declared edges) were removed regardless of theoretical elegance. The resulting system is intentionally simple: two layers (experience storage and fact extraction), hybrid search, and supersession tracking. These features add measurable value in the categories where they directly apply: supersession tracking for knowledge updates, session summaries for multi-session counting.

The benchmark finding. A controlled Oracle experiment (running the same synthesis model with perfect retrieval) scores 93.4%, revealing that the architectural contribution is +1.2 points (not statistically significant at n=500). This is not a limitation unique to Reverie: Mastra OM adds just +1.8 over its Oracle baseline with GPT-4o. LongMemEval's archives (∼115K tokens) fit comfortably within modern context windows, and strong synthesis models can compensate for architectural limitations. The benchmark is approaching saturation as a differentiator of memory architectures.

The path forward. The field needs evaluation at scales where architecture genuinely matters: hundreds of sessions, thousands of queries, archives exceeding context limits. We are developing a benchmark targeting these conditions, where retrieval quality becomes the bottleneck and architectural choices (graph-based association, vocabulary bridging, temporal indexing) become decisive.

References

Barnes, T. (2025). Observational Memory: 95% on LongMemEval. *Mastra Research.* `https://mastra.ai/research/observational-memory`

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

Emergence AI. (2025). SOTA on LongMemEval with RAG. *Blog post.* `https://www.emergence.ai/blog/sota-on-longmemeval-with-rag`

Hu, Y., Wang, Y., & McAuley, J. (2025). Evaluating Memory in LLM Agents via Incremental Multi-Turn Interactions. *arXiv:2507.05257.*

Liu, N., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2023). Lost in the Middle: How Language Models Use Long Contexts. *arXiv:2307.03172.*

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

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S.G., Stoica, I., & Gonzalez, J.E. (2023). MemGPT: Towards LLMs as Operating Systems. *arXiv:2310.08560.*

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

Reimers, N. & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of EMNLP 2019. arXiv:1908.10084.*

Supermemory. (2025). Supermemory is the New State-of-the-Art in Agent Memory. `https://supermemory.ai/research`

Vectorize. (2025). Introducing Hindsight: Agent Memory That Works Like Human Memory. `https://vectorize.io/blog/introducing-hindsight-agent-memory-that-works-like-human-memory`

Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.W., & Yu, D. (2024). LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. *arXiv:2410.10813.* ICLR 2025.

A What We Tried and Cut

The current architecture is the result of empirical pruning. Several components were built, tested, and removed. We document these to save others the same experiments and to demonstrate the build-test-prune methodology.

A.1 Association Layer (Built, Tested, Disabled)

An association layer using entity co-reference edges (entity $\leftrightarrow$ fact, bidirectional) and experiential edges (fact $\leftrightarrow$ fact, requiring both session co-occurrence and retrieval co-activation). Retrieval used a Neumann series with lateral inhibition for graph-based signal propagation. **Disabled because:** On LongMemEval, consistently net-negative. Vector search and keyword matching already achieve near-perfect recall at LME archive sizes ($\sim$40 sessions). Bridged nodes added tangentially-related fragments that degraded synthesis precision. Tested with unrestricted bridging, relevance-filtered bridging (cosine threshold), and entity traversal for counting queries; all were net-negative or showed zero effect. Additionally, LME's one-query-per-archive structure means experiential edges never form (they require ≥ 2 retrieval co-activations). The association layer may prove valuable at scale with larger archives and repeated queries, where vocabulary-gap bridging becomes necessary.

Dependent layers removed. Two further layers depended on the association layer. An *abstraction layer* extracted behavioral patterns from association clusters via Haiku (e.g., "this person approaches hobbies competitively"); it was removed because its upstream clusters never materialized. An *identity layer* maintained a persistent self-model and enriched L2 embeddings with identity context; it was removed for the same reason, and its contextual embedding enrichment was independently tested and found to degrade retrieval (Appendix A.3).

Prediction layer removed. A fourth layer used prediction error (expected vs. actual activation patterns) to weight encoding priority; "surprising" inputs received higher encoding weight. Removed because "remember everything uniformly" consistently outperformed selective encoding.

A.2 LLM-Declared Edges (v0.x)

Used LLM to declare semantic/taxonomic relationships at write time ("Ahmed IS_FRIEND_OF user", "chess IS_HOBBY_OF user"). **Replaced because:** Hallucinated relationships, inconsistent schema. Taxonomy is subjective: the LLM invented edges that did not reflect the user's actual framing. Cosine-threshold edges (connect nodes with sim > 0.6) produced dense, noisy graphs where cascade BFS reached nearly every node in 2–3 hops, making graph structure meaningless. Note: the system *does* retain LLM-declared edges for one specific purpose: supersession detection (Section 3.5), where the LLM evaluates whether a new fact logically updates an old one. This is a ground-truth evaluation ("does fact B replace fact A?") with a correct answer, unlike taxonomic classification which is inherently subjective. The distinction is between LLM-as-taxonomist (unreliable) and LLM-as-logical-evaluator (reliable).

A.3 Other Negative Results

Contextual embedding enrichment. Embedding facts with entity context appended (e.g., "User plays chess; Ahmed, bullet chess, cafe") perturbed L2 search rankings, causing regressions on questions the baseline already answered correctly. Plain text embeddings produced the 94.6% result.

Weight decay. Multiplicative decay per consolidation cycle ($w \leftarrow w \times (1 - r)$, prune below 0.3) was net destructive across all benchmarks, permanently deleting information relevant to rare queries.

Full-history synthesis in layered path. Switching the layered retrieval path to chronological full-text synthesis (instead of fragment-based) regressed from 86% to 84%, losing benefits of fragment labeling (supersession markers, event dates).