Reverie: A Layered Cortex 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 five-layer memory architecture that progressively abstracts raw conversational experiences into extracted facts, earned associations, behavioral patterns, and a persistent identity model. Two of the five layers (L1 experience storage, L2 extracted facts) drive the reported result; the remaining three require multi-query interaction that LongMemEval's single-query-per-instance structure cannot provide (Section 7). 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 a prediction layer, weight decay, and LLM-declared edges) were removed when they degraded performance.

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 memory architecture organized as a five-layer cortex inspired by the general principle of hierarchical abstraction in biological memory systems, though we make no claims of neuroscientific fidelity. The layers progressively abstract raw conversational experiences, from verbatim episode storage through extracted declarative facts, empirically earned associations, behavioral pattern abstractions, to a persistent identity model. The architecture is governed by four design principles that emerged from extensive empirical testing (Section 3.1).

Our contributions are:

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

layer, weight decay, LLM-declared edges, and resonance retrieval) 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 **five-layer cortex architecture**, of which two layers (L1, L2) are validated by LongMemEval and produce the 94.6% result.
- **Contextual embedding text**, which resolves a high encoding failure rate observed in flat-embedding approaches by enriching fact embeddings with identity context. This mechanism depends on L5 and is not exercised by LongMemEval (Section 7).
- **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.

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.3). Reverie's association edges are designed to emerge from empirical co-retrieval patterns rather than declared relationships, though this mechanism has not yet been validated at scale (Section 7).

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 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 progressive abstraction, (2) explicit supersession detection for knowledge updates, and (3) a unified architecture that scales from context-window recall for small archives to cortex-based retrieval for archives exceeding context limits. Reverie achieves state-of-the-art-class results while maintaining architectural properties (non-lossy storage, explicit knowledge update tracking) that become increasingly valuable as archive scale grows beyond what current benchmarks test.

3 Architecture

Reverie is organized as a five-layer cortex with bidirectional signal flow. Information flows upward for progressive abstraction and downward for contextualization. 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 storage that is lossless with respect to its input: the system stores every stimulus without lossy compression, though completeness depends on the integration layer delivering all relevant conversational turns.
2. **Earned associations over declared edges.** Prior approaches using LLM-declared relationships ("Ahmed IS_FRIEND_OF user") produced unreliable, hallucinated graph structures. Reverie's association edges emerge only through repeated co-retrieval: if "chess" and "Ahmed" are consistently retrieved together, an association edge forms. The graph reflects actual usage patterns, not LLM opinions.
3. **Abstractions expand queries, never compete with results.** Pattern nodes ("this person gravitates toward competitive activities") are excluded from retrieval result sets. They serve only as context for the synthesis LLM and as query expansion hints. Early versions that included abstraction nodes in results saw them flood out concrete facts.
4. **Fast ingestion, expensive consolidation.** The `perceive` operation stores raw text with no embedding computation ($\sim$3 ms). All LLM calls, embedding, fact extraction, supersession detection, and pattern mining occur during offline `consolidate`, which can run asynchronously between sessions.

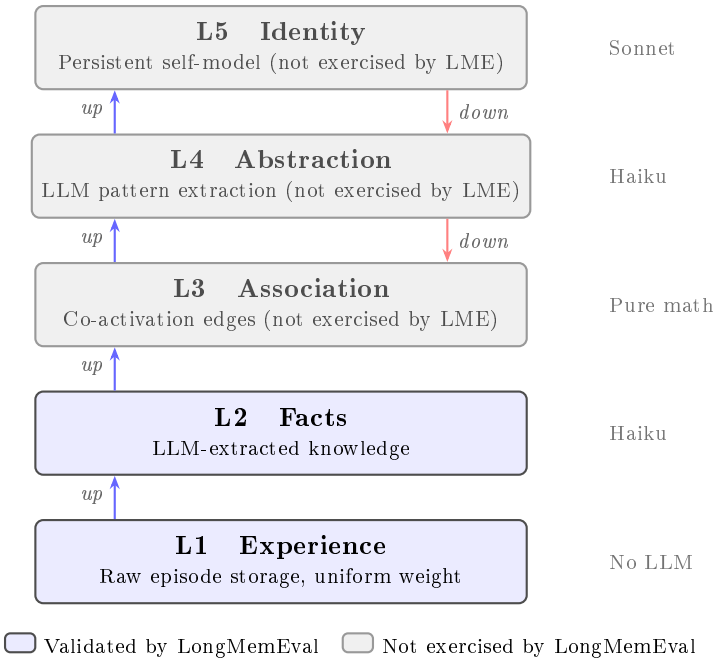

Figure 1: Reverie's five-layer cortex with bidirectional signal flow. Blue arrows (left) carry abstraction signals upward; red arrows (right) carry contextualization signals downward. L5 and L4 also send context directly to L2 (skip connections not shown for clarity). L1 receives no downward signals. Shading distinguishes layers validated by LongMemEval (L1, L2) from those that cannot activate under the benchmark's single-query-per-instance structure (L3, L4, L5).

3.2 Data Model

The system operates on four core types:

Nodes represent information at any 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 with contextual embedding text when L5 identity context is available (Section 3.6). L4 nodes store behavioral patterns. L5 maintains a single active identity model.

Edges encode relationships between nodes. Six edge types capture the architecture's structure: `co_activation` (L3, earned through repeated co-retrieval), `extracted_from` (L2→L1, provenance), `supersedes` (L2→L2, knowledge updates), `abstracted_from` (L4→L2, pattern provenance), `contextualizes` (L5→lower layers), and `world_knowledge` (seeded associations).

Signals carry information between layers during processing, flowing either upward (abstraction) or downward (contextualization).

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

3.3 Layer Details

L3, L4, and L5 are implemented but unvalidated; see Section 7.

Layer 1: Experience. Raw episode storage. Stores conversational turns verbatim with uniform weight. During consolidation, 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 deferred entirely to consolidation (no LLM calls during ingestion). Each fact is embedded with contextual text drawn from the L5 identity model when available (Section 3.6). L2 also performs supersession detection (Section 3.5).

Layer 3: Association. Co-activation edges emerge when nodes are repeatedly retrieved together (minimum 3 co-occurrences). Edges strengthen with continued co-retrieval and decay when inactive. During retrieval, L3 expands seed results by following earned edges, bridging concepts like "chess" and "Ahmed" that are semantically distant but experientially linked. No LLM involvement.

Layer 4: Abstraction. Extracts behavioral patterns from L3 clusters via Haiku (e.g., "this person approaches hobbies competitively"). L4 nodes are excluded from retrieval results and serve only as query expansion context and synthesis hints.

Layer 5: Identity. A persistent self-model updated glacially via Sonnet. Updates only every 3rd consolidation cycle and only when 5+ L4 abstractions are available (requiring L3 clusters, which in turn require multi-query interaction). Produces a comprehensive description of the user's characteristics, preferences, and behavioral patterns. This identity text enriches L2 fact embeddings (via downward contextualizing signals) and provides synthesis context during recall. Only one identity node is active at a time; updates replace rather than accumulate.

3.4 Signal Flow

Perceive (fast path, ∼3 ms): L5 sends cached context signals downward if available (computed during the last consolidation); otherwise no downward signal is sent. L1 stores the raw experience with `embedding=None`. No LLM calls, no embedding, no new 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 (contextual when L5 identity context is available), and runs supersession detection. L3, L4, and L5 run but are not exercised by LongMemEval (Section 7). Then L5 sends context signals downward for the next cycle.

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, identity context (when available), and supersession metadata. For large archives, cortex retrieval uses hybrid vector+keyword search across layers, graph expansion, context injection, supersession chain following, and fragment-based LLM synthesis.

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.

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 Contextual Embedding

A critical innovation that resolved the encoding failure problem. In the predecessor system (v0.x), facts were embedded as raw text: "chess" produced a generic embedding that failed to match queries about "Ahmed" despite their experiential connection. The majority of test queries failed to retrieve the correct fact during development testing.

The fix: embed facts with identity context appended. "chess" becomes "chess — Identity context: competitive player who enjoys bullet format with friend Ahmed." This produces embeddings that capture relational context, dramatically improving retrieval recall. The contextual text is drawn from the L5 identity model (when available) or from the most recent L5 context signal.

Cold start. Before an L5 identity model exists, L2 facts are embedded without identity context, as plain text identical to a flat-embedding baseline. L5 requires 5+ L4 abstractions to update (Section 7), so contextual embedding enrichment is only available in deployments with sufficient multi-query interaction to activate the L3→L4→L5 chain. On LongMemEval, all facts are embedded as plain text. Historical L2 nodes are *not* retroactively re-embedded when L5 eventually activates (see Section 7, "Contextual embedding staleness").

3.7 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.

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; also pattern extraction when L4 active)
- **Synthesis model:** Claude Sonnet 4.6 (memory synthesis; also identity updates when L5 active)
- **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: Anthropic API defaults (not explicitly set). 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 cortex 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 cortex 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 cortex 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").

4.4 Cost and Latency

Operation	Latency	Notes
`perceive`	∼3 ms	No LLM calls, no embedding
`consolidate`	∼25–45 s/session	L2 extraction + supersession detection
`recall`	∼3–8 s	Varies with archive size and path

Estimated cost per question on LongMemEval: ∼\$0.10 (dominated by Sonnet synthesis and Haiku extraction calls). Total evaluation cost for the full n=500 run: approximately \$50. We are not aware of comparable cost/latency data published by other systems on the leaderboard, so we do not attempt a cost comparison.

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 (cortex 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. However, the direction is consistent 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 (cortex baseline)	Cortex retrieval path, hybrid search	76%
v2	+ Session summaries, supersession chains, wider top-K	86%
v3	Switched to chronological synthesis (regressed; see A.7)	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 and are included only to show directional improvement trends.

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.*

Cortex vs. Flat Search. On our internal benchmark (15 sessions, 20 queries), flat hybrid search (BM25 + cosine) scored 82.3%. The full cortex scored 86.1%, a +3.8 point improvement. The cortex's advantage was concentrated on abstraction queries (+50%) and cross-session recall (+12%). Note: the internal benchmark uses a persistent cortex with multiple queries, unlike LongMemEval, enabling L3/L4 to contribute.

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

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

Category	Context-Window	Cortex (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 cortex vs. context-window comparison. Note: n=50 balanced sample; category-level margins should be interpreted directionally given the small per-category counts (12–13). The context-window baseline was run at v5; the cortex column is v6. Both use the same Sonnet version and GPT-4o judge.

The cortex 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 cortex 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 architectural 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.

Synthesis ceiling. We tested several retrieval augmentations (session expansion, temporal range retrieval, pre-computed date arithmetic, pre-counting candidate lists). All uniformly degraded performance (94.6% → 93.6%), consistent with context pollution: adding marginally-relevant content hurt precision without helping reasoning. An estimated 60–70% of the remaining 5.4% failure rate reflects synthesis reasoning errors (date arithmetic, multi-hop counting) rather than retrieval gaps. Genuine retrieval gaps remain (fragment coverage and vocabulary bridging), but the system is approaching retrieval saturation at this benchmark scale.

7 Limitations and Future Work

Unvalidated Layers (L3, L4, L5). L3, L4, and L5 contribute zero to the LongMemEval score by construction. LongMemEval uses 495 unique conversation histories across 500 questions, meaning each question builds a fresh cortex instance with a single query. L3 co-activation edges require repeated co-retrieval (minimum 3 occurrences); with one query per cortex, no edges are ever created. L4 abstractions require L3 clusters that never materialize. L5 identity updates require 5+ L4 abstractions, so L5 also never activates. This dependency chain (L3→L4→L5) means contextual embedding enrichment (Section 3.6), which depends on L5 identity context, is also inactive on this benchmark; all L2 facts are embedded as plain text. Additionally, each question's history is approximately 115K tokens, well within modern context window limits, so the cortex retrieval path is never strictly necessary; our cortex-only results demonstrate competitiveness despite this disadvantage. The 94.6% result is driven entirely by L1 and L2.

L3 validation plan. The most important unvalidated claim is that L3 association edges improve retrieval quality for large archives. We plan to develop a synthetic benchmark with 200+ sessions and multiple queries per cortex instance, where L3 edges accumulate and their contribution can be measured through ablation. Multi-session counting failures (where items described with different vocabulary resist both vector similarity and keyword matching) are the primary motivation; association edges would bridge vocabulary gaps through co-retrieval history.

L4 on probation. L4 abstraction patterns are theoretically useful for query expansion and synthesis context, but we cannot yet demonstrate a significant improvement attributable to L4. It is possible that the synthesis LLM can infer these patterns on the fly from sufficient raw facts, making pre-computed abstractions redundant. L4 will be retained pending scale testing. In an early ablation, including L4 nodes in retrieval results (rather than restricting them to query expansion and LLM context) caused them to flood top-K results, displacing concrete facts; restricting L4 to query-expansion-space was a +4 point improvement on the internal benchmark.

Forgetting mechanism. Node weight decay is disabled (all rates 0.0); only L3 edge decay remains active (0.1 per cycle, pruning unused associations). 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 (Section 4.4). 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.

Contextual embedding staleness. L2 fact embeddings are generated with the L5 identity context at creation time. When L5 updates (e.g., the user shifts from competitive to casual chess), historical L2 embeddings retain the stale identity context. The system does not re-embed historical facts on L5 updates. At small scale this has negligible impact, but at large scale the vector space may drift from the current identity model. Re-embedding historical nodes during L5 updates is a potential optimization.

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. Its design is empirically grounded: components that improved benchmarks were kept; components that degraded them (prediction-based encoding, weight decay, LLM-declared edges, and resonance-based retrieval) were removed regardless of theoretical elegance. The architecture's 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. Reverie's most important untested capabilities require multi-query persistent cortex instances that LongMemEval's structure cannot provide (Section 7). Validating these at scale is the key open question.

References

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

Chhikara, P., Khullar, A., Ardalani, N., & Cai, 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`

Hu, Y., Wang, Y., & McAuley, J. (2025). MemoryAgentBench: 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., Fang, V., Patil, S.G., Lin, K., Wooders, S., & 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: Atomic Memory Extraction for AI Agents. `https://supermemory.ai`

Vectorize. (2025). Hindsight: Biomimetic Memory for AI Agents. `https://vectorize.io`

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.

A.1 Layer 6: Prediction (Retired)

Maintained prediction weights for each layer. On perceive, computed prediction error by comparing expected vs. actual activation patterns. High prediction error meant "surprising" input deserving higher encoding weight. **Removed because:** (1) Fast-path perceive stores `embedding=None`, making prediction error always default (0.5). (2) Ablation showed prediction-driven weight adjustments were net destructive, with nodes receiving random boosts/penalties based on noise. (3) Zero impact on any benchmark. Core insight: "remember everything uniformly" beats "decide what to remember."

A.2 Decay (Disabled)

Multiplicative weight decay per consolidation cycle: $w \leftarrow w \times (1 - r)$. Nodes below 0.3 pruned. **Disabled because:** Ablation showed decay was net destructive across all benchmarks. It permanently deleted information relevant to rare queries. At <10K nodes, there is no retrieval noise problem to solve.

A.3 LLM-Declared Edges (v0.x)

Used LLM to declare relationships at write time ("Ahmed IS_FRIEND_OF user"). **Replaced because:** Hallucinated relationships, inconsistent schema. Cosine-threshold edges (connect nodes with sim > 0.6) produced dense, noisy graphs. Replaced by L3 co-activation: edges earned through repeated co-retrieval, not declared.

A.4 Resonance Retrieval (v0.x)

Cascade BFS through the graph: start at seed nodes, walk edges, accumulate activation scores across hops. Achieved R=0.858 on internal benchmark. **Replaced because:** The "bicycle wheel problem" (BFS through dense co-activation graphs reached nearly every node in 2–3 hops), making graph structure meaningless. Replaced by hybrid search + L3 expansion as supplement.

A.5 Organism Architecture (Pre-Cortex)

Modeled memory as a biological organism with ODE-based "heartbeat" dynamics. Nodes had activation levels governed by differential equations with resonance and interference patterns. **Abandoned because:** Mathematically interesting but practically useless; continuous dynamics added complexity without improving retrieval.

A.6 Naive Embedding (v0.x)

Embedded facts without context. "Chess" embedded as just "chess." The majority of test queries failed to retrieve the correct fact, observed during development testing on the predecessor system. The generic embedding failed to match queries about experientially-linked concepts. Fixed by contextual embedding text (Section 3.6).

A.7 Synthesize-from-History in Cortex Path (v3)

For small archives, switched the cortex path to chronological full-text synthesis instead of fragment-based synthesis. **Regressed from 86% to 84%.** The chronological format lost benefits of fragment labeling (supersession markers, event dates, layer labels). Reverted to fragment-based synthesis with small-archive expansion.