LENS Benchmark: Preliminary Results

Why Simple Retrieval Beats Complex Memory Architectures
at Longitudinal Evidence Synthesis

Mark Lubin `mark@synix.dev`

February 24, 2026

Abstract

LENS (Longitudinal Evidence-backed Narrative Signals) tests whether LLM agent memory systems can synthesize conclusions from evidence scattered across many sequential episodes. We evaluate 8 memory architectures across 6 domains, 2 context budgets, and 120 episodes per run (90 scored runs total). The central finding: **only simple BM25+embedding retrieval statistically outperforms a naive LLM that reads all evidence in context.** Knowledge graphs, agent memory, vector stores with LLM extraction, and progressive summarization all either tie or lose to the naive baseline. This report presents the full ranking, scaling behavior, and a canonical scenario trace explaining why complex architectures fail.

1. Experimental Setup

Task. An LLM agent ingests 120 sequential episodes — 30 signal episodes encoding a domain-specific storyline plus 90 format-matched distractors — then answers checkpoint questions requiring synthesis across multiple episodes. No single episode answers any question; the benchmark is broken if it does.

Domains (6). Cascading infrastructure failure, insider threat detection, environmental contamination, biotech clinical trial, IT infrastructure scaling, and financial forensics.

Systems (8).

1. **sqlite-chunked-hybrid** — BM25 (FTS5) + cosine embedding, RRF fusion, no preprocessing
2. **cognee** — Knowledge graph (entity extraction, Kuzu + LanceDB)
3. **graphiti** — Temporal knowledge graph (FalkorDB)
4. **mem0-raw** — Vector store (raw episode embeddings)
5. **letta** — Agent-managed memory (Letta server, tool-based retrieval)
6. **letta-sleepy** — Letta + sleep-time consolidation
7. **compaction** — Progressive LLM summarization (rolling summary)
8. **null** — No memory (control)

Budgets. 8K and 16K cumulative result tokens per checkpoint.

Agent LLM. Cerebras GPT-OSS-120B. **Judge LLM.** Qwen3-235B (Together AI, pairwise fact comparison).

Scoring. Three-tier composite: tier-1 mechanical metrics (evidence grounding, fact recall, evidence coverage, budget compliance, citation coverage), tier-2 LLM judge (answer quality, insight depth, reasoning quality), tier-3 differential (naive baseline advantage). Hard gate: evidence grounding ≥ 0.5.

2. Overall Rankings

Table 1: Composite score across 6 scopes, 16k budget (95% bootstrap CI). "*" = significantly different from null ($p < 0.05$, Wilcoxon signed-rank).

System	Mean	95% CI	N	vs. null
sqlite-chunked-hybrid	**0.454**	[0.406, 0.502]	6	*
cognee	0.421	[0.397, 0.446]	6	*
graphiti	0.393	[0.220, 0.491]	3	
mem0-raw	0.330	[0.265, 0.388]	6	*
letta	0.327	[0.258, 0.399]	6	*
letta-sleepy	0.322	[0.268, 0.381]	6	*
compaction	0.245	[0.137, 0.328]	6	
null	0.000	[0.000, 0.000]	6	

Only chunked-hybrid's naive baseline advantage CI is entirely above 0.5, meaning it is the only system that reliably beats a context-stuffed LLM. All others either tie (cognee, graphiti) or lose (mem0, letta, letta-sleepy, compaction).

Concordance. Kendall's $W = 0.755$ across all paired comparisons; 12 of 28 pairwise tests significant at $p < 0.05$.

Budget effect. 16K significantly outperforms 8K ($p = 0.016$, paired Wilcoxon across all adapters).

Caveat. Post-hoc audit found that the LLM judge silently failed on 21/90 runs (23%), primarily affecting cognee (12/12 judge failures) and letta variants (4/12 each). Failed judgments defaulted to TIE, inflating scores. Cognee's true ranking cannot be determined without re-scoring.

3. Cross-Domain Consistency

Table 2: Composite score: Adapter × Scope (16k budget). "—" = did not finish.

System	S01	S02	S03	S04	S05	S06	Mean
chunked-hybrid	0.424	0.482	0.381	0.535	0.520	0.386	**0.454**
cognee	0.438	0.402	0.374	0.471	0.418	0.423	0.421
graphiti	0.491	0.220	—	—	—	0.467	0.393
mem0-raw	0.345	0.320	0.186	0.419	0.407	0.299	0.330
letta	0.288	0.338	0.215	0.455	0.424	0.239	0.327
letta-sleepy	0.300	0.313	0.252	0.451	0.377	0.240	0.322
compaction	0.339	0.000	0.198	0.364	0.309	0.259	0.245
null	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Chunked-hybrid ranks first or second in 5 of 6 scopes. Graphiti DNF'd on scopes 03–05 due to entity extraction timeouts (~600 LLM calls per scope). Compaction catastrophically fails on scope 02 (composite = 0.000).

4. Scaling Behavior

How do systems behave as data grows from episode 1 to 120? We extract per-checkpoint metrics averaged across all scored scopes for each adapter.

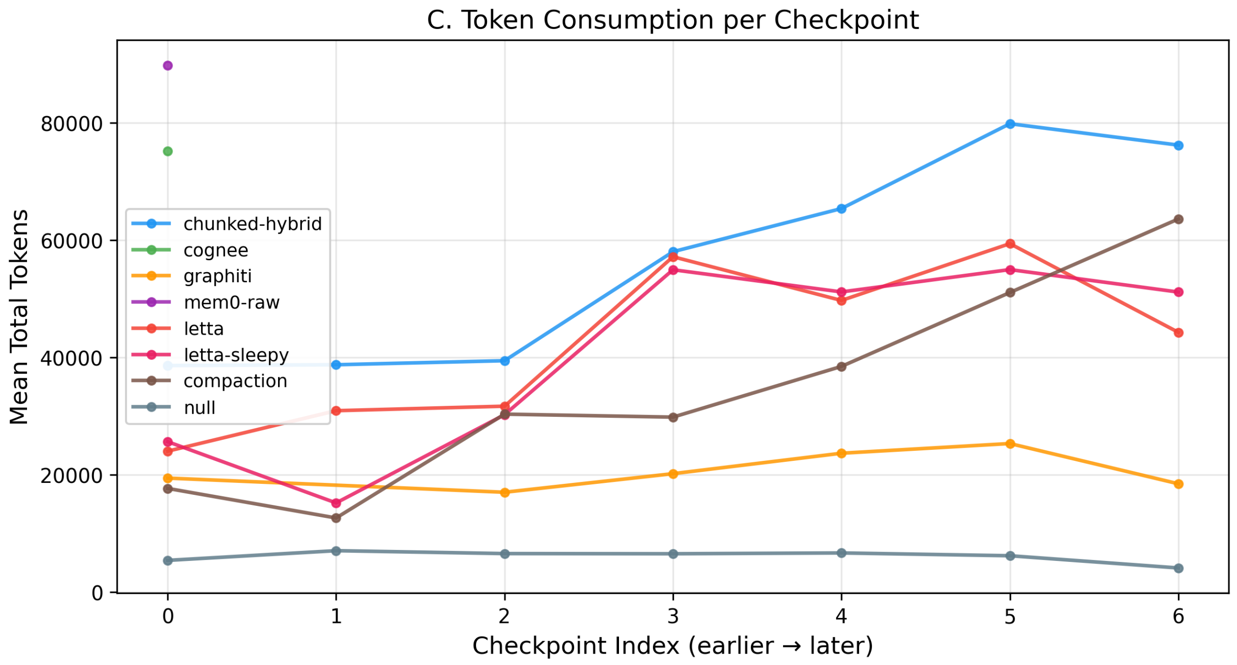


Figure 1: Token consumption per question grows with corpus size. Chunked-hybrid rises linearly (bounded by retrieval limit). Letta/letta-sleepy/compaction climb sharply as search reformulation burns tokens. Null stays flat (no memory to search).

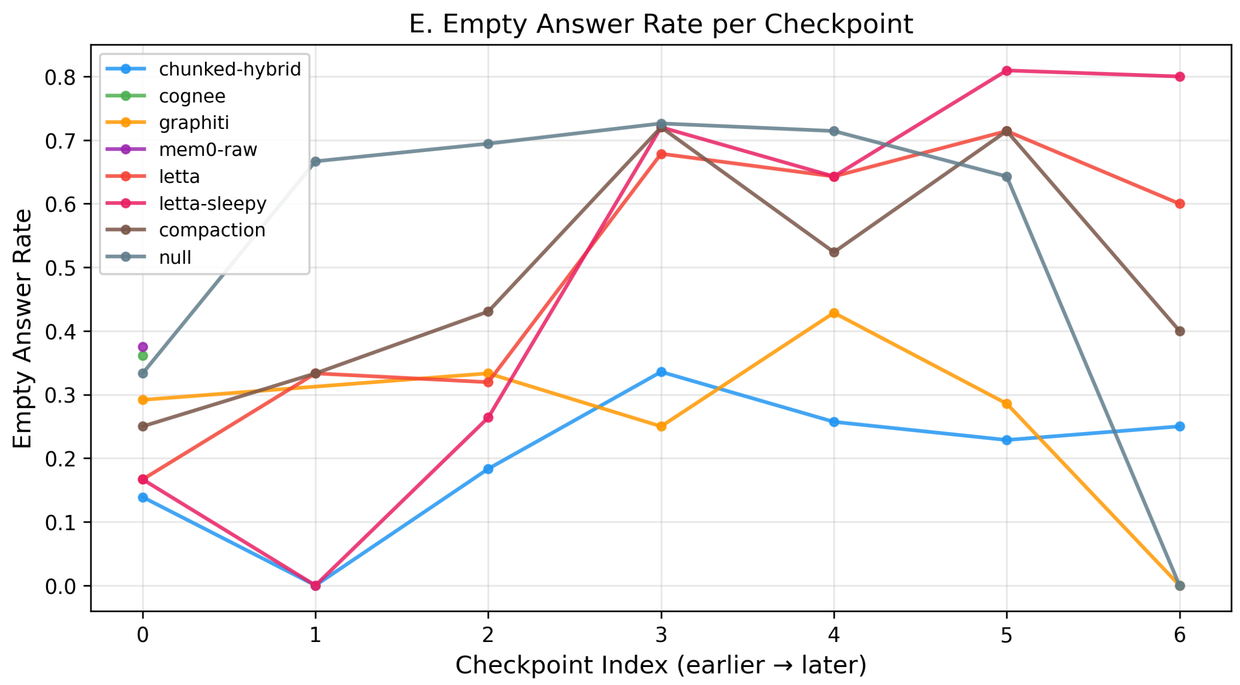
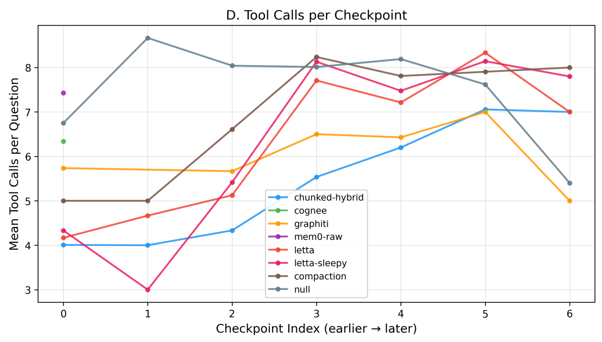
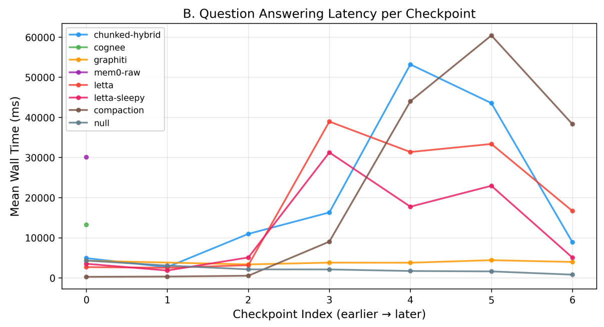


Figure 2: Empty answer rate by checkpoint. Chunked-hybrid maintains the lowest rate (14–33%). Letta and compaction degrade to 60–80% as the corpus grows and retrieval becomes unreliable.



(a) Tool calls per question



(b) Wall-clock time per question

Figure 3: Operational scaling. Chunked-hybrid starts at ∼4 calls and rises slowly. Other systems converge toward the 10-call turn limit. Compaction's wall time spikes at later checkpoints due to growing summary re-generation.

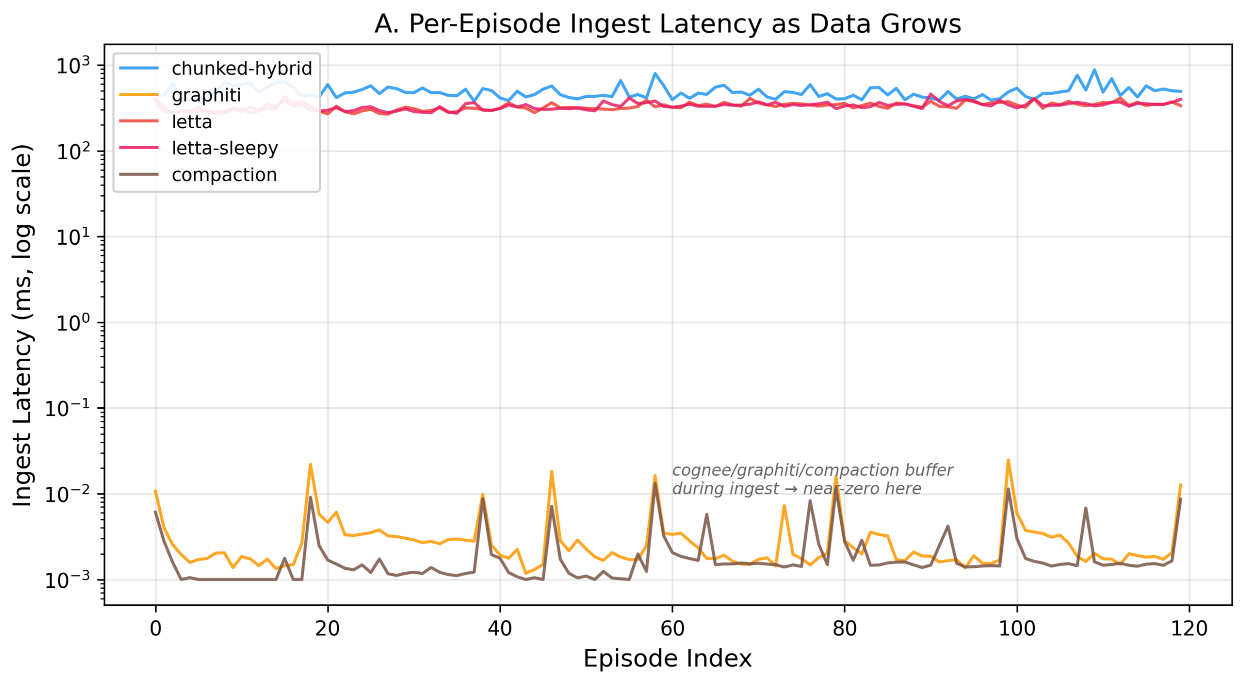


Figure 4: Per-episode ingest latency (log scale). Letta and chunked-hybrid process episodes during ingest (∼300–700ms). Graphiti and compaction buffer during ingest and do real work in `prepare()` — their near-zero ingest times are misleading without that context.

Key observation. Token consumption (Figure 1) is the real scaling signal. When retrieval fails, agents don't stop — they reformulate and retry. Each retry consumes tokens for the query, the empty response, and the agent's internal reasoning. This *search thrashing* explains why complex architectures burn 4–8× more tokens than chunked-hybrid for the same questions.

5. Canonical Scenario Trace

To make the failure modes concrete, we trace one question through all systems.

5.1. The Question

Scope 04 (environmental drift): 6 water quality stations along a river, 30 days of monitoring + 90 distractors.

Q3 (longitudinal): *"What is the source of the chromium contamination and how can you determine its location?"*

Ground truth: An unpermitted discharge pipe between stations WQ-02 and WQ-03 at river-mile 18.6. The spatial gradient proves this: WQ-01/02 at baseline, WQ-03 at peak (132 µg/L), downstream stations show dilution. Agricultural runoff explains turbidity but not chromium.

Key evidence: Episode 025 — a field inspection note recording the pipe discovery.

5.2. Results

Table 3: Canonical scenario: Chromium source question (ed04_q03, checkpoint 99, 16k budget).

System	Calls	Tokens	Time (ms)	Answer	Pipe?	RM 18.6?
chunked-hybrid	**3**	**22,308**	**2,679**	**1,642 ch**	✓	✓
compaction	4	35,774	62,337	2,607 ch	×	×
null	6	4,537	1,409	212 ch	×	×
letta	8	74,722	126,527	2,567 ch	×	×
letta-sleepy	9	88,547	69,344	*empty*	×	×
cognee	9	157,187	19,723	*empty*	×	×
mem0-raw	9	180,102	7,183	2,293 ch	×	×

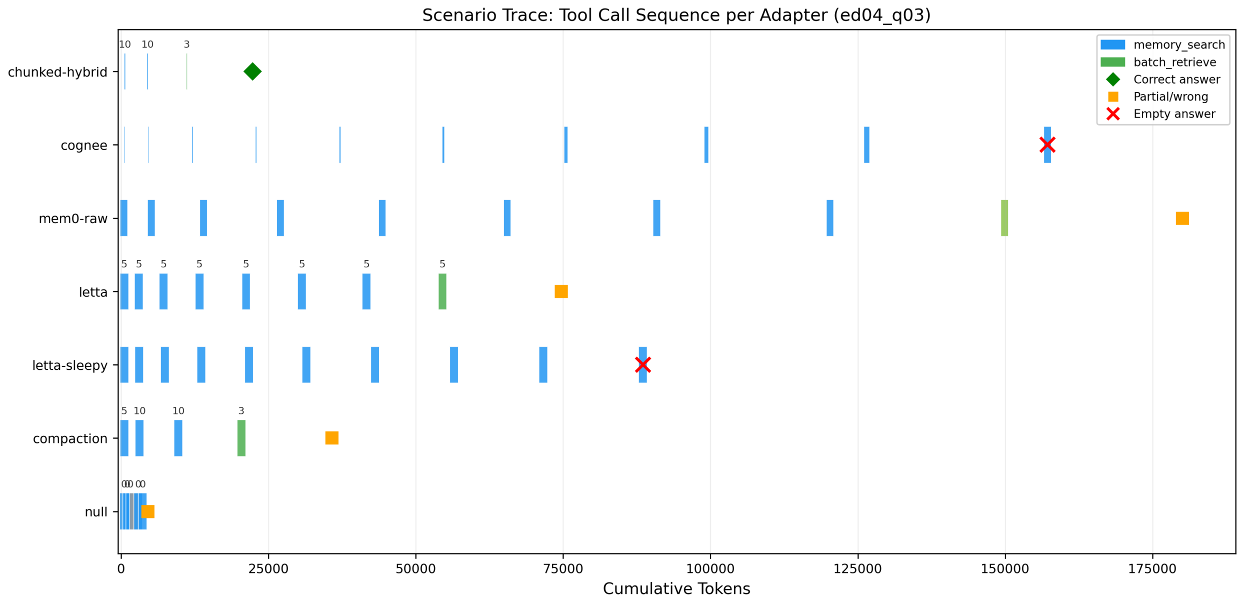


Figure 5: Tool-call sequence per adapter for the chromium source question. Each blue block is a `memory_search`, each green block is a `batch_retrieve`. Green diamond = correct answer, orange square = partial/wrong, red X = empty answer. X-axis shows cumulative tokens consumed.

5.3. What Each System Did

Chunked-hybrid (3 calls, correct): Two `memory_search` queries find episode 025 at rank 5 via BM25 keyword overlap ("source" + "chromium" + "discharge"). One `batch_retrieve` returns the complete episode text. The agent reads "unpermitted discharge pipe identified between WQ-02 and WQ-03 at RM 18.6" and synthesizes a grounded answer with the spatial gradient data. Three valid citations.

Letta-sleepy (9 calls, empty): Nine progressively longer `memory_search` queries — "chromium contamination source," "chromium source contamination analysis," "chromium source contamination upstream industrial" — each returning nothing actionable. The agent burns 88K tokens (5.5× the 16K budget) reformulating queries before hitting the 10-turn limit with no answer.

Cognee (9 calls, empty): Same search-thrashing pattern. The knowledge graph indexed entities and relationships from structured monitoring data but didn't capture the narrative field note containing the answer. The discharge pipe is described in natural language prose, not as a graph node.

Mem0-raw (9 calls, wrong): Vector search retrieves episodes 018 and 026 (escalation data) but misses episode 025. The agent identifies the wrong station and mislocates the source — semantically similar episodes aren't the right episodes.

Compaction (4 calls, partial): Retrieves episodes 001–003 (early baseline). Its rolling summary absorbed episode 025 into a generalized statement. The specific detail — one sentence about a pipe at a specific river mile — was compressed away.

6. Why Complex Systems Fail

Three patterns explain the rankings.

6.1. The Lossy Transformation Trap

Every architecture except chunked-hybrid introduces a lossy transformation between ingestion and retrieval:

Table 4: Lossy transformations introduced by each architecture.

System	Transformation	What's Lost
chunked-hybrid	None (raw text indexed)	Nothing
cognee	Entity extraction → graph	Temporal ordering, narrative details
mem0-raw	Vector embedding	Keyword-level specificity
letta	Agent-managed memory	Search reliability (stochastic retrieval)
letta-sleepy	Agent memory + consolidation	Same as letta + compression noise
compaction	Rolling LLM summary	Numeric precision, per-episode detail

Longitudinal synthesis requires *all* the details: exact values, temporal ordering, narrative context. When any of these are discarded by an intermediate LLM — whether summarizing, extracting entities, or re-encoding into embeddings — the answering agent works from a degraded input.

6.2. Search Thrashing

When retrieval fails, the agent reformulates and retries. Each retry costs tokens. Systems with unreliable retrieval (letta, letta-sleepy, cognee) burn 4–8× more tokens than chunked-hybrid on the same questions (Figure 1). This directly degrades performance: the token budget is exhausted on failed searches rather than answer synthesis.

6.3. The Keyword Advantage

Episode 025 contains "unpermitted discharge pipe" and the question asks about "the source of chromium contamination." BM25 matches these through shared terms. Embedding similarity requires the model to have learned that "discharge pipe" is semantically close to "contamination source" — a reasonable inference, but one that competes with 119 other episodes also discussing contamination. For sparse, specific facts, exact keyword match beats semantic similarity.

7. Summary

1. **Simple retrieval wins.** BM25 + embedding hybrid is the only system that statistically outperforms a naive context-stuffed baseline ($p < 0.001$).
2. **No system exceeds 0.50 composite.** The benchmark is hard — even the winner answers fewer than half the questions correctly.
3. **Complexity hurts.** Knowledge graphs, agent memory, and progressive summarization all introduce lossy transformations that destroy the details longitudinal synthesis requires.
4. **Token consumption is the scaling signal.** Systems that can't find evidence in 1–2 queries burn their budgets on search thrashing, leaving nothing for answer synthesis.
5. **16K significantly outperforms 8K** ($p = 0.016$), confirming that budget constraints are binding.

These results suggest that current memory architectures are optimized for the wrong problem. They trade fidelity for abstraction, assuming the agent needs pre-digested knowledge. For longitudinal synthesis — where the signal is sparse, specific, and distributed — the agent needs raw evidence and reliable retrieval. Everything else is noise.