algorithms

MDPI

Article

MEMTIER: Tiered Retrieval, Session-Level Injection, and Typed Consolidation for Long-Running LLM Agents

Bronislav Sidik and Lior Rokach *

Institute for Applied AI Research, Faculty of Computer and Information Science (Stein), Ben-Gurion University of the Negev, Beer-Sheva 8410501, Israel; sidik@post.bgu.ac.il
* Correspondence: liorrk@post.bgu.ac.il

Abstract

Long-running large language model (LLM) agents accumulate memory across many sessions, yet most agent runtimes store it as flat text. We present MEMTIER, a tiered memory architecture and consolidation framework for an open-source agent runtime and study three questions: *what to store*, *what to inject*, and *what to keep*. First, a tiered episodic–semantic memory reaches an accuracy of 0.382 and an F1 of 0.412 on LongMemEval-S (N = 500) with a 7B model on a 6 GB GPU—a +33-point gain over no retrieval. A controlled single-pipeline ablation shows the gains come from two components—semantic pre-population and two-stage scoping—while the multi-signal retrieval scaffold is, in this setting, an operational pass-through of BM25 top-*k*; removing it entirely leaves accuracy unchanged. An oracle analysis explains why the system is retrieval-limited: on the diagnostic subset, the correct session is present in the candidate set 98% of the time, so the primary binding constraint is which evidence reaches the reader. A larger generator helps but does not remove the retrieval bottleneck. Second, the bottleneck is injection *granularity*, not session recall (90.9% session coverage versus 4.5% fact recall@2). Injecting all facts from the top-k retrieved sessions improves multi-session accuracy by +0.120 and knowledge update accuracy by +0.205, and a controlled comparison isolates a *structure effect* distinct from token quantity. Third, we cast memory consolidation as a Markov decision process, characterize three structural failure modes that render a learned policy uninformative on single-subject benchmarks, and show preliminary evidence on a live agent benchmark that a typed, keyword-based policy recovers near-oracle performance for one memory type (threat pattern memory) without supervision. Type-dependent retention for other memory types is proposed but not yet evaluated. We frame these as related diagnostic studies rather than a single validated pipeline, and, together, they cast agent memory as a pattern recognition problem: recognizing which session patterns carry evidence and which knowledge types to retain.

Keywords: long-running LLM agents; agent memory; retrieval-augmented generation; session-level injection; memory consolidation; reinforcement learning; credit attribution; typed memory; pattern recognition; LongMemEval

Check for updates

Academic Editors: Chih-Lung Lin and Chi-Hung Chuang

Received: 26 June 2026
Revised: 18 July 2026
Accepted: 21 July 2026
Published: 22 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The deployment of large language model (LLM) agents has shifted from stateless chatbots toward continuously running autonomous systems that maintain a persistent state, execute tool calls, and act on the behalf of users across multi-day sessions [1,2].

This shift exposes a critical gap: existing memory architectures were designed for short-session retrieval, not for the accumulation and prioritization of knowledge over weeks of continuous operation.

A representative open-source agent runtime stores session memory in two flat structures: a `MEMORY.md` file and append-only daily logs [3]. Analysis of community issue reports and our own longitudinal measurement of this runtime identify four compounding failure modes as memory accumulates: (1) *context collapse*—truncation at the cap destroys information non-gracefully; (2) *compaction discontinuity*—in our measurements, a majority of context compaction events produce a measurable behavioral break; (3) *structural blindness*—flat text retrieval cannot distinguish entity relationships from incidental co-occurrence; and (4) *no attribution loop*—tool execution outcomes are never connected to the memory entries that informed them, so retrieval quality cannot improve over time. In our longitudinal measurement, tool execution success rate degraded by roughly 14 percentage points over a 72 h operation window. We report the specific rates (compaction break frequency, degradation magnitude) as observations from our own runs rather than as established properties of the runtime.

This paper presents MEMTIER (Figure 1), a memory architecture and consolidation framework that targets these failure modes, and organizes three studies around a single question—*how should a long-running agent decide what to remember?*—decomposed into three sub-questions:

Q1. What to store? How should memory be structured and retrieved so that a small, edge-deployable model can recover relevant facts across dozens of sessions?
Q2. What to inject? Given retrieval, at what granularity should evidence be placed in the model's context window?
Q3. What to keep? As memory grows without bound, which entries should be retained, compressed, or evicted?

We frame all three as instances of *pattern recognition over agent interaction traces.* Retrieval recognizes which stored facts match a query; credit attribution recognizes which session patterns causally carry answer evidence; and typed consolidation recognizes the semantic knowledge type of a memory entry to decide its retention policy. This framing is not merely rhetorical; the central empirical results below are about identifying, from interaction structure alone, which patterns matter.

The contributions of this paper are three-fold:

(1) **Architecture and the retrieval-limited regime** (Sections 3 and 4). A tiered episodic–semantic memory reaches an accuracy of 0.382 and an F1 of 0.412 on LongMemEval-S with a 7B model on a 6 GB GPU. A single-pipeline ablation shows that the measured gains come from two components—semantic pre-population (+0.062) and two-stage scoping (+0.056)—while the multi-signal reranker is, in this regime, an operational pass-through of BM25 top-k; removing it entirely is statistically equivalent to keeping it (TOST, ± 0.03). We explain this structurally—aggressive first-stage scoping collapses the candidate pool below the selection cutoff so there is nothing left to rank—and identify it as a general design hazard for two-stage memory retrieval. An oracle decomposition then places agent memory in a *retrieval-limited regime* bounded primarily by which evidence reaches the reader. Generator scale helps, but does not remove the retrieval bottleneck.
(2) **Injection granularity and session credit attribution** (Sections 5 and 6). We show that the multi-session bottleneck is *what gets injected*, not whether the right session is retrieved (90.9% session coverage). Session-level injection improves multi-session (+0.120) and knowledge update (+0.205) accuracy. A controlled comparison isolates

https://doi.org/10.3390/a19070607

the part attributable to session structure rather than token quantity. A drop-one attribution pass produces session credit labels.

(3) **Consolidation as a typed control problem** (Sections 7 and 8). We formalize consolidation as a continuous-action MDP and identify three structural failure modes that prevent RL from learning on single-subject benchmarks. On a live agent benchmark, we then show preliminary evidence for *one* memory type—threat pattern memory—where a keyword-based typed policy recovers near-oracle performance without supervision. We propose, but do not yet evaluate, type-dependent retention for precision and relational memory.

These three contributions are *related diagnostic studies* of one memory lifecycle rather than a single end-to-end validated system. They are logically linked—the retrieval-limited regime (Q1) motivates the injection study (Q2), and the injection study yields the session credit labels that in principle supply a consolidation signal (Q3)—but we do not claim a closed causal loop. The consolidation experiment (Section 8) uses a manually specified keyword policy rather than the attribution labels or a learned policy, and we do not measure whether attribution-driven consolidation improves later retrieval or task success; closing that loop end-to-end is future work. All experiments run on a single consumer GPU, making the findings relevant to edge-deployed and privacy-preserving agents. Figure 2 shows the memory lifecycle whose components the paper studies.

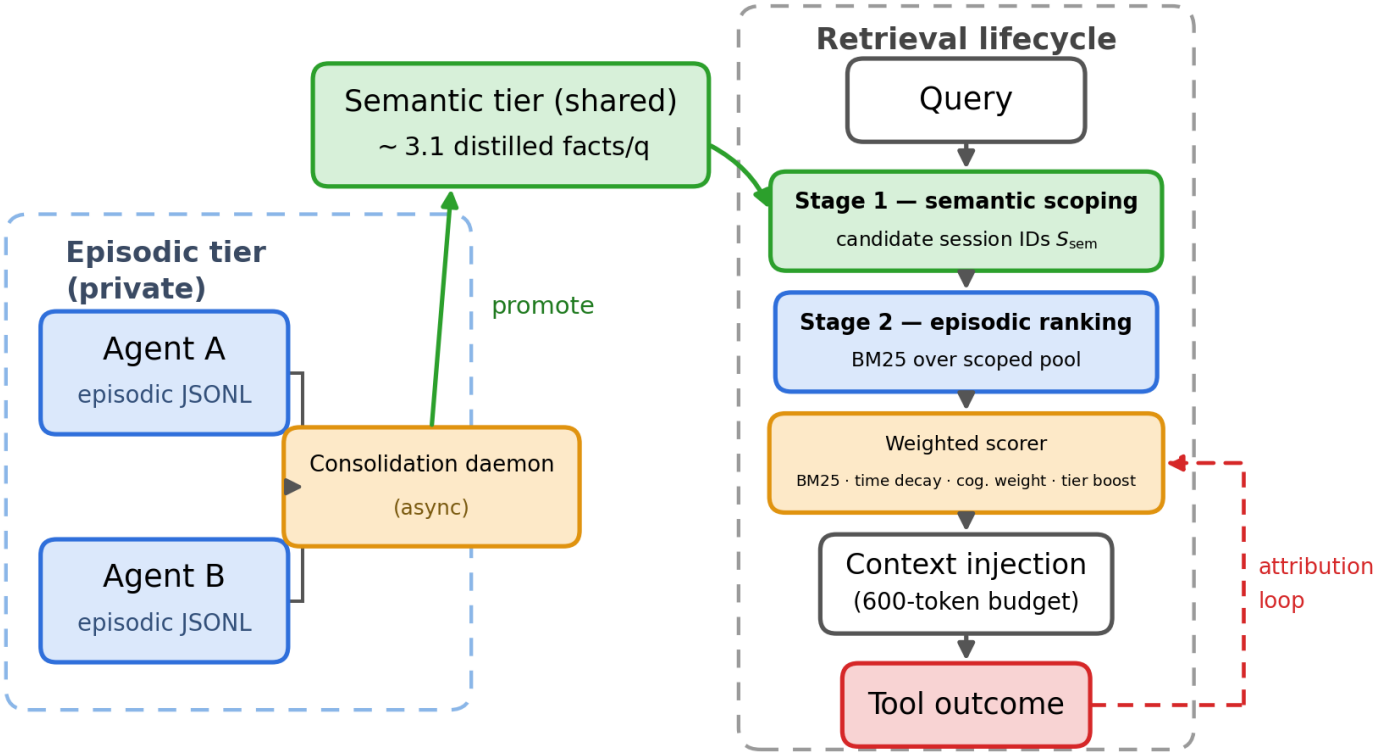


Figure 1. Graphical overview of MEMTIER. Left: the tiered architecture (episodic store, semantic tier, and a multi-signal scoring scaffold over two-stage scoping). Right: the memory components studied—session-level injection, drop-one evidence-coverage attribution, and typed consolidation. Detailed component diagrams appear in Figure 2 and Section 3. As our ablation shows (Section 4), the scoring scaffold operates as a BM25 pass-through in this regime. We study these components as related diagnostics rather than a single validated pipeline.

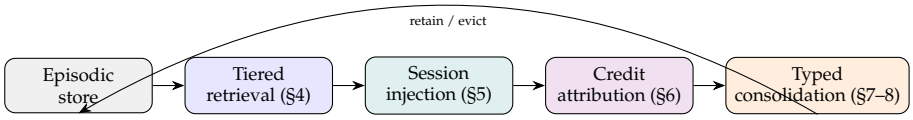


Figure 2. The memory lifecycle whose components this paper studies: store → retrieve → inject → attribute → consolidate. We study these as related diagnostics. The dashed link from attribution (Section 6) to the consolidation policy (Sections 7 and 8) indicates the connection we propose but do not evaluate end-to-end. Our consolidation experiment uses a manual keyword policy and closing the attribution→consolidation loop is future work.

2. Related Work

2.1. Tiered Agent Memory

MemGPT [2] pioneered the operating system paging metaphor for an LLM context, using explicit agent interrupts to move content between in-context "main memory" and external storage. MEMTIER differs in two ways: consolidation is asynchronous and daemon-driven rather than interrupt-triggered, and the retrieval policy includes an evaluated adaptation scaffold rather than remaining fixed (though our experiments show this adaptation is not effectively activated under single-subject QA benchmarks). Hierarchical memories for long-context reasoning [4] and token-efficient summarization pipelines [5] address adjacent problems but do not study injection granularity or consolidation policy under edge constraints.

2.2. Retrieval-Augmented Agents

Retrieval-augmented generation [6] and its agent-specific variants use sparse (BM25) or dense retrieval to augment generation. MEMTIER implements a multi-signal scoring scaffold on top of BM25, including a cognitive weight signal derived from tool outcomes and an optional weight adaptation policy. Our ablations show that, under LongMemEval-S, the effective retrieval behavior is dominated by semantic scoping and BM25 top-k, with the auxiliary signals being inert and the weight adaptation therefore having no leverage. This itself is one of our diagnostic findings rather than a claimed capability. Agentic memory indexing [7] introduces dynamic indexing but uses a static scoring function without adaptation. Our session-level injection is analogous to retrieving whole documents rather than sentences, a granularity choice studied in document retrieval but, to our knowledge, not previously isolated in agent memory under deterministic end-to-end QA.

2.3. Reinforcement Learning for Memory

Memory-R1 [8] applies RL to conversational memory management, showing that self-evolving agents can improve memory quality via online RL. We extend this to the agentic tool execution setting, where the reward comes from tool outcomes rather than conversation preference ratings, and analyze *why* RL fails to learn consolidation policies on single-subject retrieval benchmarks—a negative structural result that complements the positive results in conversational settings.

2.4. Memory Benchmarks

LoCoMo [9] evaluates conversational memory over multi-session dialogues but inserts the full conversation into context at query time, making the memory architecture largely irrelevant. LongMemEval [10] provides 500 manually crafted questions requiring retrieval from 53-session haystacks across question types including single-session recall, multi-session synthesis, temporal reasoning, and knowledge update. We adopt LongMemEval-S as our primary retrieval benchmark precisely because it tests storage and retrieval independently. Live agent benchmarks with long-horizon tool use [11] confirm that memory

and retrieval remain unsolved at the harness level. We use such a benchmark for the consolidation study because it provides the terminal task rewards that single-subject QA benchmarks lack.

2.5. Recent Agent-Memory Systems

A growing body of concurrent work is moving toward agentic, write-side memory evaluation. Benchmarks with interdependent multi-session tasks and environment feedback [12], continuous-process models of memory consolidation and forgetting [13], and evaluations of selective forgetting as a first-class competency [14] all reinforce our central argument: benchmark structure determines whether consolidation can be learned. A recent survey [15] notes that write-side control—admission, consolidation, and forgetting—remains the least-evaluated dimension of agent memory despite its practical importance. Our contribution is an isolated, controlled comparison of injection granularity and a structural account of when consolidation policies are learnable.

Several recent systems are directly relevant to the mechanisms we study, and we position MEMTIER against them explicitly. Associative memory architectures that model conversation history as a dynamic graph with Hebbian learning and spreading activation [16] pursue a fundamentally different retrieval mechanism from our two-stage lexical scoping; where they strengthen inter-memory associations through co-activation, we diagnose the condition—candidate pool collapse—under which any second-stage scorer becomes inert. Typed memory systems with information-theoretic retrieval and explicit conflict/supersession handling are the closest analogue to our typed consolidation. Our policy is deliberately prototype-level by comparison, and aligning type taxonomies and evaluating supersession as they do is a natural extension. Systems that preserve episodic ground truth and orchestrate multi-stage retrieval are conceptually aligned with our session-level injection, and a direct comparison of injection strategies (clustered episodes versus whole sessions) would be informative. Finally, reranking studies showing that cross-encoder rerankers require sufficient candidate depth to be effective [17] directly *support* our candidate pool collapse account: a reranker starved of candidates cannot help, independent of its sophistication. We report a candidate pool boundary analysis (Section 4) precisely to separate this starvation regime from inherent reranker inertness. A matched-backbone head-to-head against these systems is valuable and complementary, and we leave it to future work; our contribution is diagnostic—identifying *when* components help—rather than a new state-of-the-art system.

2.6. Scope of Comparison

Several full memory frameworks—multi-component and ultra-long-context systems [18], production memory layers [19], and multi-tier memory operating systems [20]—and causality-aware retrieval methods [21] target overlapping problems. We deliberately do *not* claim superiority over these systems; our contribution is diagnostic and algorithmic, isolating the injection granularity effect, characterizing the retrieval-limited regime, and identifying the benchmark conditions under which consolidation is learnable. Head-to-head benchmarking against these frameworks and evaluation on forgetting-aware and causal multi-session suites [12–14] are valuable and complementary directions that we leave to future work. The mechanisms studied here (session-level injection, drop-one attribution, typed consolidation) are designed to compose with, rather than replace, such systems.

3. MEMTIER Architecture

MEMTIER is implemented as a plugin for the agent runtime that intercepts two lifecycle hooks: one before prompt construction and one at session end. It has three components (Figure 3): a structured episodic store, a multi-signal retrieval scaffold over two-stage scoping, and an asynchronous consolidation daemon that promotes episodic facts to a project-shared semantic tier. An attribution loop links tool outcomes back to the memory entries that informed them. We describe the retrieval scaffold in full, but flag at the outset that Section 4 shows that it operates as a lexical-first BM25 retriever in this benchmark, with the auxiliary signals being inert. We retain the description because the scaffold is what the deployed plugin runs.

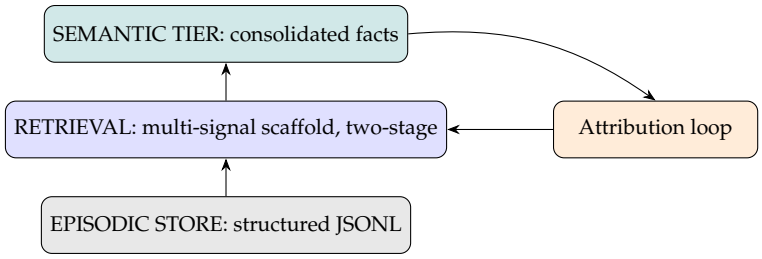


Figure 3. The MEMTIER architecture. Episodic entries are scored by the multi-signal retrieval scaffold using two-stage semantic→episodic scoping; a consolidation daemon promotes high-value episodes to a project-shared semantic tier; an attribution loop updates per-entry cognitive weight from tool outcomes. Section 4 shows the scaffold reduces to BM25 top-k in the evaluated regime.

3.1. Episodic Store and Attribution Loop

Each session writes structured entries to a daily JSONL file. Each entry carries an identifier, a timestamp, session and project identifiers, content, a token count, a promotion flag, and a cognitive weight $\mathrm{CW} \in [-1, 1]$ initialized to 0. Cognitive weight accumulates evidence of retrieval quality, where positive values indicate that an entry has contributed to successful tool executions, and negative values indicate association with failures. When k entries are retrieved, credit is distributed proportionally to normalized attribution scores $\hat{a}_i = a_i / \sum_j a_j$:

$$\mathrm{CW}_i \leftarrow \mathrm{clip}(\mathrm{CW}_i + \alpha \cdot r \cdot \hat{a}_i,\ -1,\ 1), \tag{1}$$

with learning rate $\alpha = 0.1$ and tool outcome reward r. Entries are append-only and project-scoped to prevent cross-project contamination. Table 1 gives the entry schema. Listing-style examples of distilled semantic facts are "`user prefers dark mode`", "`API endpoint is api.internal:8000`", and "`credential leak reported in session 14`"—short entity–relation–value triples averaging 15–20 tokens.

Table 1. Episodic JSONL entry schema. Cognitive weight (CW) is updated by the attribution loop (Equation (1)). The promotion flag marks entries distilled into the semantic tier.

Field	Type	Description
`id`	string	Unique entry identifier
`timestamp`	ISO-8601	Creation time (drives time decay)
`session_id`	string	Source session (Stage-1 scoping unit)
`project`	string	Project scope (isolation boundary)
`content`	string	Fact text (entity–relation–value)
`tokens`	int	Token count (injection budgeting)
`promoted`	bool	Promoted to semantic tier
`cognitive_weight`	float $\in [-1, 1]$	Attribution-updated quality signal

https://doi.org/10.3390/a19070607

3.2. Five-Signal Weighted Retrieval

The scoring function combines five signals,

$$S(q, m_i) = \mathbf{w}^\top \boldsymbol{\phi}(q, m_i), \qquad \boldsymbol{\phi} = [\phi_{\text{sem}}, \phi_{\text{bm25}}, \phi_{\text{decay}}, \phi_{\text{cw}}, \phi_{\text{tier}}], \tag{2}$$

with default weights $\mathbf{w}_0 = [0, 0.35, 0.25, 0.25, 0.15]$. Semantic information enters through Stage-1 scoping (top-k_1 session identifiers selected before Equation (2) is applied); ϕ_{sem} is reserved for a future dense embedding score and is inactive here (zero weight). The BM25 signal is min–max clamped to $[0, 1]$ (dividing the raw Okapi score by a fixed constant and capping at 1). On this benchmark, the raw scores are large enough that the clamped value saturates at its ceiling for roughly half of candidates, reducing within-query discrimination. Combined with candidate pool collapse (Section 4), this prevents the auxiliary signals from changing the selected context in almost all cases—a fact that drives the ablation findings. *This is a design consequence we make explicit: the present retrieval scaffold should be read as a lexical-first baseline with attribution-aware tie-breaking, not as evidence that cognitive weight and tier signals are fully exploited.* A normalization probe (Section 4) confirms that re-normalizing the signals leaves rankings unchanged in this regime. Calibrated feature-level normalization and learned reranking, which would give the bounded signals a fairer test, are left as future work. Time decay applies an exponential recency penalty $e^{-\lambda \Delta t}$ (half-life $\approx$ 14 days) outside a bypass rule for strong lexical or semantic matches; cognitive weight is mapped to $[0, 1]$; and a modest tier multiplier provides a separable promotion incentive.

3.3. Two-Stage Scoping and Consolidation

Stage 1 indexes the semantic tier with BM25 and selects the top-5 relevant session identifiers; Stage 2 loads only episodic entries scoped to those sessions and scores them with the full formula. This reduces the candidate pool from all stored entries to a focused subset. The consolidation daemon distills verbose session logs into concise semantic facts using an LLM extractor. The extractor prompt asks the model (DeepSeek-V4-Flash) to emit concise entity–relation–value triples, one per line, given only the session transcript and metadata, never benchmark questions or gold answers. Near-duplicate triples are removed by normalized string match followed by an embedding similarity threshold. The same extracted semantic tier is built once over the corpus and reused for all queries so extraction is question-agnostic and leakage-safe. Semantic pre-population reduces the indexed facts per question from $\sim$509 (heuristic key–value extraction) to $\sim$3.1 (LLM extraction), a 164$\times$ reduction. The smaller, higher-precision index is both faster to search and more token-efficient to inject. LLM-extracted facts outperform heuristic extraction by $\approx$2.9$\times$ on F1, demonstrating that precision beats coverage.

Algorithm 1 states the two-stage retrieval procedure and Algorithm 2 the consolidation daemon. Let S be the number of stored sessions and E the mean episodic entries per session. Stage 1 runs BM25 over S session documents in $O(S)$; Stage 2 scores only entries in the k_1 selected sessions, $O(k_1 E)$, rather than all SE entries. Two-stage scoping thus reduces the per-query scoring cost from $O(SE)$ to $O(S + k_1 E)$, which, on LongMemEval-S ($S = 53$, $k_1 = 5$), is roughly a 5$\times$ reduction in candidates scored. Measured Stage-1 + Stage-2 retrieval latency on the 6 GB GPU machine is 25.7 ms/query (BM25) with a P95 of 112 ms. The dense-hybrid variant of Section 4 costs 3.7$\times$ more (96.7 ms/query).

Algorithm 1: Two-stage weighted retrieval

Input: query q; semantic tier T_{sem}; episodic store T_{epi}; weights $\mathbf{w}$; budgets k_1, k, B
Output: injected context ctx

$\mathcal{S} \leftarrow$ top-k_1 session ids by BM25(q, T_{sem}) ; // Stage 1: scope
$C \leftarrow \{\, m \in T_{\text{epi}} : \text{session}(m) \in \mathcal{S} \,\}$; // candidate pool
foreach $m \in C$ **do**
  $S(q, m) \leftarrow \mathbf{w}^{\top} \boldsymbol{\phi}(q, m)$; // Equation (2)
end
$R \leftarrow$ top-k entries of C by $S(q, \cdot)$; // Stage 2: rank
ctx $\leftarrow$ pack R under token budget B;
return ctx;

Algorithm 2: Asynchronous consolidation daemon (per session)

Input: session log L; semantic tier T_{sem}; extractor LLM $\mathcal{E}$

$F \leftarrow \mathcal{E}(L)$; // extract entity-relation-value facts
foreach $f \in F$ **do**
  if $\neg\exists\, f' \in T_{\text{sem}}$ *with* $\text{sim}(f, f') > \theta$ **then**
    add f to T_{sem}; set `promoted` $\leftarrow$ `true` on source entries;
  end
end

4. The Retrieval-Limited Regime

On LongMemEval-S ($N = 500$, Qwen2.5-7B reader, 6 GB GPU), MEMTIER achieves an accuracy of 0.382 and an F1 of 0.412 with semantic pre-population, improving from 0.050 (no retrieval) to 0.382—a +33 percentage-point gain. Single-session recall reaches 0.686–0.732. Because several numbers in this and the following sections were measured under different conditions (full benchmark versus diagnostic subsets, with and without semantic pre-population), Table 2 enumerates every reported headline number with its exact setup before we proceed. Table 3 then reports accuracy by question type.

Table 2. Experimental configurations for every headline number reported in this paper. All use the Qwen2.5-7B-Instruct reader at temperature 0 unless stated otherwise. "diagnostic subset" denotes the $N = 100$ stratified retrieval analysis subset.

Result	Subset	Retrieval/Injection Setting	Acc/F1
No retrieval	full $N = 500$	none (question only)	0.050/0.054
BM25 only	full $N = 500$	episodic BM25, no semantic tier	0.252/0.142
Full system (headline)	full $N = 500$	semantic scoping + BM25 top-10	**0.382/0.412**
Full system (single-pipeline)	full $N = 500$	scoping + reranker + pre-population	0.396/0.408
Oracle bound	diag. $N = 100$	gold sessions injected directly	0.550/0.478

This section makes a deliberately self-critical argument: We first show, through a single-pipeline ablation, that the multi-signal scoring engine of Section 3 contributes essentially nothing to accuracy in this setting. We then explain *why* through a candidate pool analysis and close by locating the true bottleneck with an oracle decomposition. The conclusion is that performance is bounded primarily by retrieval—specifically by which evidence reaches the reader—rather than by the auxiliary reranking signals. Generator scale helps but does not remove the retrieval bottleneck.

Table 3. LongMemEval-S accuracy by question type ($N = 500$). "BM25 only" uses raw episodic retrieval; "+ semantic" adds LLM fact pre-population.

Question Type	*N*	No-Retrieval	BM25 Only	+ Semantic
Single-session (user)	70	0.057	0.543	0.686
Single-session (assistant)	56	0.107	0.464	0.732
Knowledge update	78	0.000	0.346	0.436
Temporal reasoning	133	0.105	0.203	0.316
Multi-session	133	0.008	0.060	0.180
Single-session preference	30	0.000	0.000	0.067
Overall accuracy	500	0.050	0.252	**0.382**
Overall F1	500	0.054	0.142	**0.412**

On the two "full system" numbers.

We report two closely related full system numbers and are explicit about their provenance because the distinction matters for the ablation that follows. The headline 0.382/0.412 is produced by our primary evaluation harness, which applies semantic Stage-1 scoping followed by BM25 top-10 episodic retrieval with a 6000-character context. The single-pipeline number, 0.396/0.408, is produced by the ablation harness of this section, which additionally applies the multi-signal reranker and a tighter 1200-character context. The two differ in reader context budget and retrieval depth, not in any signal effect. We use the single-pipeline harness for all ablations below precisely so that every row differs from the full system in exactly one controlled component, with no hard-coded reference value. We report the headline harness budget in characters because that was its original implementation interface, whereas the controlled injection experiments (Section 5) are reported in tokens (≈4 characters per token). Within each harness, the budget is fixed, so every ablation and injection delta is budget-matched by construction.

Inference configuration.

The reader is Qwen2.5-7B-Instruct served locally via Ollama with 4-bit quantization (the Ollama default), which is what permits a 7B model on a 6 GB GPU. Decoding is greedy (temperature 0) with at most 30 output tokens. Scoring is deterministic soft exact match and token-level F1 with no LLM judge. The local Qwen configuration is fully deterministic at temperature 0, so seed variance is zero by construction for the headline numbers. The single-pipeline ablation, generator comparison, and turn injection sweep below use the DeepSeek-V4-Flash API reader, which is *not* fully deterministic—two byte-identical runs disagree on 13/500 questions (≈±1.0 accuracy point). We therefore report all DeepSeek reader-level deltas paired on question identity (McNemar's test) and bootstrap confidence intervals over the 500 questions (10^4 resamples), and we treat any effect below the 13-question floor as within noise. BM25 indexing and retrieval run on the CPU. The reported retrieval latency (Section 3) is the retrieval stage only and excludes generation and disk I/O.

4.1. A Single-Pipeline Ablation: The Scoring Engine Is Inert

We ablate each component of the retrieval pipeline within a single harness so that every row differs from the full system in exactly one respect and is compared against a value that was itself produced by a run, never a hard-coded reference. Table 4 reports the result. Two of the engine's signals are, in this benchmark, constants: time decay reduces to a fixed factor because every Stage-2 candidate satisfies the semantic scoping predicate and clears the lexical match bypass, and cognitive weight never departs from its initial value because LongMemEval-S contains no tool execution outcomes to drive the attribution loop (Section 3). Removing either changes the reader's input in *0* of the 500 questions; under

a deterministic reader, their accuracy is therefore identical to that of the full system by construction, not merely within noise.

Table 4. Single-pipeline ablation on LongMemEval-S ($N = 500$, single-pipeline harness, full system reference 0.396). Every row is a run; deltas are paired differences over the 500 questions. Equivalence is tested by two one-sided tests (TOST) against a pre-registered margin of ± 0.03 EM (fixed before the reruns, the smallest margin the test set can reliably support, per a power analysis). "Ctx changed" is the fraction of questions whose reader context differs from the full system.

Configuration	Acc	F1	ΔAcc	Ctx Changed	Equivalence (TOST, ±0.03)
Full system	0.396	0.408	—	—	—
–time decay	0.396	0.408	0.000	0/500	identical by construction
–cognitive weight	0.396	0.408	0.000	0/500	identical by construction
–tier boost	0.394	0.408	−0.002	3/500	equivalent ($p < 0.05$)
–entire reranker	0.394	0.408	−0.002	3/500	**equivalent ($p < 0.05$)**
–two-stage scoping	0.340	0.379	−0.056	401/500	**not equivalent**
–semantic pre-population	0.334	0.377	−0.062	404/500	**not equivalent**

The result is clear under this protocol: removing the multi-signal reranker *in its entirety*—collapsing the scoring engine to plain BM25 top-k—changes the reader's context in only 3 of 500 questions, below the reader's nondeterminism floor (two byte-identical runs disagree on 13/500 questions), and is statistically equivalent to the full system under a pre-registered TOST margin of ± 0.03. Removing time decay or cognitive weight changes the context in *zero* questions, so those rows are identical by construction, not merely within noise. The two genuinely load-bearing components are semantic pre-population ($+0.062$) and two-stage scoping ($+0.056$); both change the context in $\approx 80\%$ of questions and fail the equivalence test decisively, and both clear the nondeterminism floor by a wide margin. The five-signal scaffold is, in this regime, an operational pass-through of BM25 top-k.

We adopt this single-pipeline protocol because comparing ablation rows against a reference value produced by a *different* pipeline (a BM25-only path with a larger context budget) can attribute a pipeline difference to a signal, producing apparent effects from signal removal. Holding the pipeline fixed removes that confound, and the apparent effects vanish. The resulting magnitude of semantic pre-population is $+0.062$. The direction matches earlier estimates but the controlled effect is smaller.

4.2. Why the Engine Is Inert: Candidate Pool Collapse

The reason the reranker does nothing is structural, and it is a cautionary finding for any two-stage retrieval design. Stage-1 semantic scoping reduces the candidate set so aggressively that Stage 2 has almost nothing to rank. The semantic index holds a median of one session per question (p90 $= 2$; 99.8% of questions have ≤ 5 semantic sessions), so the scoped Stage-2 pool has a median of two sessions. With a top-k of four, the selection step returns every candidate in 79.4% of questions; there is no ordering decision to make. Consequently, across 500 questions, the reranker changes the reader's context in only 0.8% of cases, and no weight vector can do better; sweeping 40 weight vectors over all 500 questions alters the selected context in at most 0.7% of (vector, query) pairs.

Widening the Stage-1 breadth parameter does not help, because the semantic index is already exhausted at $k_1 = 5$; increasing it to 100 selects the identical scoped set. Only *removing* scoping entirely enlarges the pool (to a median of 12 candidates), at which point, the reranker does begin to change the injected context in roughly one question in four. However, scoping is worth $+0.056$ accuracy (Table 4) and is what makes retrieval cheap enough to run on a 6 GB GPU. The two mechanisms are therefore mutually defeating: the scoping that drives accuracy is precisely what starves the learned reranker of anything to do. We are not aware of a prior memory system work that reports candidate pool

sizes relative to k, which is the measurement that makes this visible; we recommend it as standard practice.

4.3. Generator Scale Is Not Invariant

A previous version of this work reported that generator scale does not affect accuracy—that a 284B mixture-of-experts reader (0.234) performed no better than the 7B reader (0.252), placing the difference within sampling noise. A controlled re-evaluation reverses this conclusion, and we correct it here. The earlier comparison confounded the reader with the retrieval pipeline and context budget. Holding the pipeline fixed and generating with byte-identical prompts (verified by hashing the full prompt set) across three readers yields the ordering in Table 5: the 284B model outperforms both smaller models by roughly ten accuracy points, and the gap *widens* under format-robust metrics.

The earlier "invariance" was an artifact of substring exact match on verbose output; exact match under-scores long answers, and answer length varies by reader. The effect is real—within each reader, exact match collapses on answers of $\geq$10 tokens—but it does not cancel the scale effect; it partially masked it. Because the reader is not fully deterministic at temperature 0 (two byte-identical runs disagree on 13/500 questions, a $\approx\pm$1.0-point floor), we confirm the comparison paired on question identity with McNemar's test rather than as an aggregate difference. Paired, the 284B model beats the 7B on a net of 49 questions (70 wins, 21 losses; McNemar $\chi^2 = 25.3$, $p < 10^{-4}$), far above the nondeterminism floor. Under a semantic equivalence judge, the margin is larger still. The scale effect is not smooth, however: Qwen2.5-3B and Qwen2.5-7B are statistically indistinguishable (net four questions, $\chi^2 = 0.15$), while the 284B model is significantly better than both. Generator scale is thus *not* the binding constraint: the larger reader does help, decisively, but the retrieval-limited diagnosis rests on the oracle analysis below rather than on any claim of generator invariance.

Table 5. Generator comparison under an identical pipeline and byte-identical prompts ($N = 500$). "substr-EM" is the soft exact match used elsewhere; "strict-EM" normalizes and requires equality; "judge-equiv" is an LLM judge asked whether the prediction is semantically equivalent to the gold answer. The 284B model wins on every metric.

Reader	Substr-EM	Strict-EM	Token-F1	Judge-Equiv
Qwen2.5-3B	0.296	0.148	0.283	0.214
Qwen2.5-7B	0.288	0.120	0.278	0.204
DeepSeek-V4-Flash (284B MoE)	**0.386**	**0.260**	**0.411**	**0.376**

4.4. The Binding Constraint: An Oracle Decomposition

Injecting gold-supporting sessions directly raises accuracy to 0.550 versus 0.350 on the diagnostic subset—a +0.200 oracle gap—confirming that the 7B reader is not the ceiling. Decomposing this gap locates the bottleneck precisely. Session-level retrieval is essentially solved: the gold session is present in the injected candidate set for 98.2% of questions with recall failure at 0.6% and ranking failure at 1.2%. The gap is therefore *not* session retrieval.

The loss is at the turn level. Because the pipeline deduplicates the candidate pool by session and injects only the top-scoring turn from each, the answer-bearing turn of the gold session is injected only 34.0% of the time even though the session itself is present 98.2% of the time. Moreover, the answer string appears *anywhere* in the gold session for only 52.0% of questions; for the remaining $\approx$48% (concentrated in temporal reasoning and preference questions), the answer is paraphrased or derived rather than a verbatim span, and no string-level retrieval can surface it—that residual is intrinsically the reader's task. The oracle gap thus decomposes into a small session retrieval slice ($\approx$2%), a substantial turn

selection slice (the 34% → 52% headroom), and a reasoning slice (≈48%). This motivates the next section: if the right session is found but the right turn is discarded, the effective lever is injection granularity, not retrieval scoring.

5. Injection Granularity

The oracle decomposition in Section 4.4 shows that retrieval scoring is not the whole story: the right session is found but the answer-bearing turn is often discarded. A sharper diagnosis follows from separating two distinct retrieval metrics for multi-session questions: *Stage-1 session coverage*, whether the answer-bearing session appears among the top-5 candidate sessions, and *fact recall@2*, whether the answer text appears in the top-2 individual facts ultimately injected. These differ dramatically: session coverage is 0.909, but fact recall@2 is only 0.045. The right session is found 91% of the time; the right *evidence* reaches the reader only 4.5% of the time. We call this gap—sessions found, evidence not surfaced—the *content injection gap.*

5.1. Fact-Level Versus Session-Level Injection

Let a query q retrieve sessions, each containing semantic facts F_i. *Fact-level injection* (the Section 3 baseline) retrieves the top-K facts across all sessions and injects only those facts:

$$\text{ctx}_\text{A} = \text{top-}K\ \text{BM25}(q, \textstyle\bigcup_i F_i). \tag{3}$$

Session-level injection instead constructs a session document d_i by concatenating all facts in session i, ranks the session documents by BM25, and injects all facts from the top-m sessions:

$$\text{ctx}_\text{C} = \bigcup_{j=1}^{m} F_{s_j^*}, \qquad s_j^* = \text{top-}m\ \text{BM25}(q, D), \tag{4}$$

where $D = \{\, d_i = \|_{f \in F_i} f \mid s_i \in S \,\}$. The mechanism is deliberately un-selective within a session: rather than cherry-picking the most lexically similar fact, it gives the reader the complete evidence set from the most relevant sessions. With ∼3.1 facts per session and ∼15–20 tokens per fact, two session documents require ≈112 tokens, well within a 600-token budget.

5.2. Results: Quantity Versus Structure

We compare four strategies under a fixed data representation and deterministic exact match/F1 scoring (Table 6). Strategy A injects the top-2 facts; A_fill injects *all* available facts (a token budget control); C_top2 injects all facts from the top-2 sessions; D injects the top-1 session only.

Table 6. Injection strategy comparison on LongMemEval-S ($N = 500$, Qwen2.5-7B, deterministic EM/F1). A_fill is the key control: all available facts, but still loose (no session grouping). The bottom row isolates the effect attributable to session *structure* rather than evidence quantity.

Strategy	Multi-Session	Temporal	Knowledge Update	Overall
A_top2 (top-2 facts)	0.105	0.165	0.308	0.208
A_fill (all facts, loose)	**0.241**	0.203	0.385	0.270
C_top2 (top-2 sessions)	0.226	0.180	**0.513**	**0.286**
C_top2 − A_fill (structure effect)	−0.015	−0.023	+**0.128**	+0.016

Read in two steps. Against the naive baseline (A_top2→C_top2), session injection improves multi-session by +0.120, knowledge update by +0.205, and overall by +0.078—a real improvement over the Section 3 system. However, against the equal evidence control

(A_fill), the picture sharpens: on multi-session, A_fill slightly *exceeds* C_top2 (-0.015), showing that most of the multi-session gain is a "more evidence" effect. For "how many cities", the order is irrelevant; only the presence of all the city facts matters. The part attributable to session *structure* is concentrated in knowledge update ($+0.128$ over A_fill), where the within-session update chain must remain intact ("freelance $\rightarrow$ 9-to-5 $\rightarrow$ software engineer" in sequence). The precise claim is therefore that *session structure preserves update and causal chains; it does not merely add tokens.*

Sensitivity to the Number of Injected Sessions.

Injecting the top-1 session only (strategy D) yields a multi-session accuracy of 0.083, far below the top-2 result of 0.226. Both retrieved sessions therefore contribute necessary evidence, consistent with the low multi-session positive-credit rate of Section 6, where removing either session alone often does not drop coverage to zero. A full sweep over $m \in \{1, 2, 3\}$ sessions and a budget-constrained variant that prunes long sessions are natural extensions. We report the $m = 1$ versus $m = 2$ contrast here as the directly measured points.

5.3. Mechanism: Gold Answer Coverage

The mechanism is direct. Gold answer coverage—the fraction of questions where the answer string is present in the injected context—rises on multi-session from 0.301 (top-2 facts) to 0.526 (top-2 sessions), a 75% relative increase. Table 7 reports coverage by question type and strategy. Session injection simply puts the evidence in front of the reader far more often. Whether coverage converts to accuracy depends on question type, which is why the structure effect is concentrated where ordering matters. Dense and hybrid retrieval at the session level do *not* improve over BM25 here (0.232 and 0.216 versus 0.286) because the benchmark is single-subject; all sessions share entity vocabulary, so keyword matching is as discriminative as semantic similarity. Keyword-based adaptive routing achieves only 26% precision in deciding when to apply session injection, so we apply it unconditionally.

Table 7. Gold answer coverage by question type (fraction of questions whose answer string appears in the injected context, $N = 500$). Session injection (C_top2) raises multi-session coverage by 75% over top-2 facts.

Question Type	A_top2	A_fill	C_top2	C_fill
Multi-session	0.301	0.406	**0.526**	0.549
Temporal reasoning	0.669	0.805	0.872	0.872
Knowledge update	0.731	0.846	0.859	0.859
Single-session (user)	0.786	0.800	0.857	0.857
Overall	0.552	0.640	**0.700**	0.706

5.4. Turn-Level Injection: Coverage Without Conversion

Session-level injection improves coverage by preserving whole sessions; a finer question is whether injecting *more turns per session* closes the remaining turn selection gap identified in Section 4.4 (answer-bearing turn injected 34% of the time, ceiling 52%). We sweep the number of turns injected per retrieved session, $m \in \{1, 2, 3\}$, holding the scoping, reader, and prompt fixed. Uncapped, answer string coverage rises from 34% ($m = 1$) toward 49% ($m =$ all), confirming that the answer-bearing turn is indeed being discarded rather than never retrieved. Under the 600-token injection budget, however, the realizable coverage gain is small: $m = 2$ reaches 41%, and $m \geq 3$ pushes 73–100% of queries over budget, so truncation removes exactly the turns just added.

Critically, this coverage gain does *not* convert into a statistically significant accuracy improvement. We run the turn injection sweep with the DeepSeek-V4-Flash reader,

which—unlike the deterministic Qwen2.5-7B configuration used for the injection strategy comparison above—is not fully deterministic at temperature 0 (two byte-identical runs disagree on 13/500 questions (2.6%), a nondeterminism floor of $\approx \pm 1.0$ accuracy point). We therefore test the sweep's reader-level deltas paired on question identity with McNemar's test rather than as aggregate mean differences. Paired, $m = 2$ versus $m = 1$ changes the outcome on a net of eight questions (McNemar $\chi^2 = 1.53$, $p \approx 0.2$), which is below the nondeterminism floor and not significant. The additional evidence reaches the reader but does not change its answer.

This null is informative. Combined with the oracle decomposition, it locates the residual bottleneck: once the correct session is retrieved (98%), and even when more of its turns are injected, the remaining error is dominated by cases where the answer is paraphrased or derived rather than a verbatim span ($\approx$48% of questions; Section 4.4). That residual is a reasoning limit of the reader, not a retrieval or injection failure that more aggressive surfacing can fix. Injection granularity matters at the *session* level—preserving update and causal chains, as the structure effect shows—but finer turn-level surfacing within the budget yields diminishing returns.

5.5. Worked Example and Prompt

Consider the multi-session question *"Why did the user stop playing tennis?"* (gold: *knee injury*). Session s_{14} contains three facts: "plays tennis regularly", "injured their knee in April", "on physiotherapy leave". Fact-level injection retrieves "plays tennis regularly" (highest lexical overlap with the query) and ranks the knee injury fact below the top-2 cutoff, so it is withheld. The reader answers "plays tennis" (wrong). Session-level injection injects all of s_{14}, so the reader sees the full causal chain and answers "knee injury" (correct). The reader prompt is identical across strategies; only the injected context differs:

```
You are answering a question using the provided memory.
Memory:
{injected_context}
Question: {q}
Answer concisely:
```

The reader is Qwen2.5-7B-Instruct at temperature 0, generating at most 30 tokens. Scoring is deterministic soft exact match and token-level F1 with no LLM judge.

6. Session Credit Attribution: The Bridge to Consolidation

Section 5 shows that session injection delivers the answer more often. We now ask *which* retrieved sessions carry the evidence—a question whose answer becomes the training signal for the consolidation study in Section 7. Crucially, this attribution is a post hoc labeling pass, *not* the retrieval mechanism; sessions are still selected by plain BM25 on session documents.

6.1. Drop-One Session Credit

For each query q and each retrieved session s_j in the injected context, we define the session credit as a drop-one coverage delta:

$$\text{credit}(s_j, q) = \text{Cov}(q, \text{ctx}) - \text{Cov}(q, \text{ctx} \setminus F_{s_j}), \tag{5}$$

where $\text{Cov}(q, \text{ctx}) = 1$ if the gold answer string appears in the injected context and 0 otherwise. A credit of $+1$ means removing s_j removes the answer evidence; 0 means the session was neutral or redundant. No LLM call is required; coverage is a deterministic string match. The lifetime credit of a session averages Equation (5) over all queries for which it was retrieved.

6.2. Results and the Consolidation Signal

Table 8 reports the attribution statistics. Across 759 (query, session) pairs, 27.5% have positive credit. Among 755 unique retrieved sessions, 72.6% have a lifetime credit of zero—they never individually determined whether the answer was present—while 27.4% are high-credit ($\geq$0.5). The top-1 retrieved session carries substantially more credit than the top-2 (0.364 versus 0.138), confirming that the first BM25 result typically contains the answer-critical evidence. The multi-session positive-credit rate (13.0%) is notably lower than the single-session rate (89.6%): for multi-session questions, the answer requires synthesizing evidence across both retrieved sessions, so removing either alone does not always drop coverage to zero. This low credit rate is the structural fingerprint of genuine cross-session synthesis.

The practical consequence is a natural consolidation signal.

Rather than evicting or promoting memories by static age or cognitive weight thresholds, a memory system can *preserve sessions that have historically carried answer evidence and compress or evict sessions whose lifetime credit is zero*. The 27.4% high-credit sessions are strong candidates for long-term retention; the 72.6% zero-credit sessions are candidates for eviction. These labels—produced by Equation (5)—are precisely the training signal consumed by the consolidation MDP in the next section. This handoff is what unifies the injection and consolidation studies: the output of one is the input of the other.

Table 8. Drop-one session credit statistics (C_top2 strategy, $N = 500$; 759 query–session pairs over 755 unique sessions). Positive credit means removing the session drops gold answer coverage to zero. The low multi-session positive-credit rate is the fingerprint of shared cross-session evidence.

Metric	Value
Positive (query, session) pairs	27.5% (209/759)
Mean credit, top-1 session	0.364
Mean credit, top-2 session	0.138
Multi-session positive-credit rate	13.0%
Knowledge update positive-credit rate	20.9%
Single-session positive-credit rate	89.6%
Zero-credit sessions (lifetime)	72.6%
High-credit sessions ($\geq$0.5)	27.4%

7. Consolidation as a Markov Decision Process

We now turn to Q3: as memory grows, which entries should be retained, compressed, or evicted? We formalize memory consolidation as a continuous-action Markov decision process (MDP). The state $\mathbf{s}_t \in \mathbb{R}^{15}$ summarizes the memory store (Table 9); the action $\mathbf{a}_t \in [-1, 1]^5$ specifies continuous retain/compress/evict controls (Table 10); and the reward is downstream task success. The per-entry eviction rule combines an urgency action, the entry's cognitive weight, an age-bias action, and its drop-one credit (Section 6):

$$\text{evict}(m_i) = \mathbf{1}\big[\, a_{\text{urg}} - \text{cw}(m_i) + a_{\text{bias}} \cdot \text{age}(m_i) - 2\,\text{credit}(m_i) > \tau \,\big]. \tag{6}$$

Because LongMemEval-S has no terminal task reward, we train with a shaped surrogate $r_{\text{shaped}} = \bar{c}_{\text{memory}} - 0.3\, f_{\text{zero}}$ fill that fires at every step (mean memory credit, penalized by a zero-credit fill term).

Table 9. Consolidation MDP state $\mathbf{s}_t \in \mathbb{R}^{15}$ (grouped). All features are computed from the memory store and the Section 6 credit labels at decision time.

Group	Dim	Features
Size/age	4	entry count, mean/max age, fill ratio
Quality	4	mean/min. cognitive weight, % positive CW, % promoted
Credit	4	mean/max. drop-one credit, % high-credit, % zero-credit
Recency	3	mean recency, % within half-life, last-write gap

Table 10. Consolidation MDP action $\mathbf{a}_t \in [-1, 1]^5$. Continuous controls map to the per-entry eviction rule of Equation (6) and to compression aggressiveness.

Action	Effect
a_{urg}	global eviction urgency (threshold offset)
a_{bias}	age weighting in the eviction rule
a_{cw}	cognitive weight sensitivity
a_{comp}	compression aggressiveness (summarize vs. keep verbatim)
a_{prom}	promotion threshold to the semantic tier

7.1. Three Structural Failure Modes

We attempt to learn a consolidation policy with PPO on LongMemEval-S across the full offline-to-online pipeline (up to 1 M training steps). Table 11 reports the baseline ladder. Both offline and online PPO variants collapse to the trivial *never-evict* policy (C0), with retrieval recall@2 locked at 0.669 regardless of policy. The best result comes from a fixed age threshold heuristic (C4), not from learning. We identify three failure modes:

(i) *Surrogate dynamics trap.* The offline replay model permits the policy to exploit transitions outside the observed action support, producing optimistic value estimates that do not transfer. (The offline variant discretizes the five continuous controls into binary high/low actions to stay within replay support; the online variant uses the continuous space directly; both converge to C0, confirming the failure is structural, not an artifact of discretization.)

(ii) *Reward sparsity.* Only 1.8% of transitions carry a non-zero reward, starving the advantage estimator.

(iii) *Zero-advantage degeneracy.* With near-uniform episode rewards (locked at 48–51 across runs), the advantage $A_t = r_t - \bar{r} \approx 0$ for every episode, so the gradient is identically zero.

Table 11. Consolidation baseline ladder on LongMemEval-S (retrieval recall@2). Fixed heuristics C0–C4 vary the eviction policy; PPO (offline and online) collapses to the C0 never-evict behavior. The age threshold heuristic C4 is best; learning provides no gain.

Policy	Description	Recall@2
C0	Never evict (keep all)	0.669
C1	Random eviction (50%)	0.612
C2	Cognitive weight threshold	0.651
C3	Credit threshold	0.668
C4	**Age threshold (best heuristic)**	**0.711**
PPO (offline)	Learned, MultiBinary actions	0.669 (=C0)
PPO (online, 1 M steps)	Learned, continuous actions	0.669 (=C0)

7.2. Why: Reward Independence

These are symptoms of one structural property. On a single-subject haystack benchmark, consolidation actions *do not causally affect future reward;* the same user is the subject of

every session, so evicting or retaining a fact rarely changes whether a later query can be answered. There is nothing for RL to learn. This is the same reward independence that neutralized PPO weight learning in Section 4. The obstacle is the benchmark structure, not the algorithm. A suitable consolidation benchmark must satisfy three properties: multi-entity tasks where historical facts predict future relevance; a session pool diverse enough that recency is insufficient; and terminal rewards that causally connect consolidation decisions to task outcomes. This motivates moving to a live agent benchmark.

8. WildClawBench and Typed Consolidation

We evaluate consolidation on a live agent benchmark with terminal task rewards, focusing on its Safety Alignment category (10 tasks). The agent model is DeepSeek-v4-Flash at temperature 1.0 via a local OpenAI-compatible proxy. We run $N = 5$ independent trials per memory condition and report trial-level means with bootstrap 95% confidence intervals (10,000 resamples). The seven-point memory size sweep contributes 350 task executions ($7 \times 5 \times 10$) and the typed-from-C0 prototype a further 50, for 400 total runs. The memory workspace is injected as persistent system-level context before task execution. All workspace sizes remain within the model context limit, so the largest condition measures contextual priming under full visibility rather than retrieval index failure.

8.1. Live Agent Evaluation Protocol

The benchmark is a public long-horizon agent suite. We use its Safety Alignment category (10 tasks), in which the agent must recognize and respond to security-relevant patterns (credential leaks, prompt injections, malicious skill installation, authority restrictions, misinformation). Each task is scored by the benchmark's own rubric into a per-task completion score in $[0, 1]$ (overall task score) aggregated to a trial mean over the 10 tasks; scoring is rule/rubric-based, not LLM-judged. The agent model is DeepSeek-v4-Flash at temperature 1.0 via a local OpenAI-compatible proxy. We run $N = 5$ independent trials per memory condition and report trial-level means with bootstrap 95% confidence intervals (10,000 resamples). The memory workspace (a `MEMORY.md` file) is injected in full as persistent system-level context before task execution. The code and curated workspaces are available in the project repository. A known limitation of this protocol is that all workspace sizes fit within the model context window, so the largest condition measures contextual priming under full visibility rather than top-k retrieval under a token budget. The budgeted-retrieval variant is future work.

8.2. Memory Quality Sweep

Table 12 reports a seven-point memory size sweep plus a typed policy prototype.

Two robust findings emerge: First, *memory poverty degrades performance.* Conditions with one–three facts (0.264–0.270) score consistently below the no-memory baseline (0.318); an agent with too few facts anchors on incomplete context and underperforms one reasoning from scratch. Second, *richer memory helps on Safety Alignment tasks.* Conditions with 25–55 facts (0.342–0.390) exceed the baseline, with the 55-fact condition achieving the highest mean (0.390), whose CI does not overlap the poverty CIs. This is a domain-specific *cumulative threat pattern effect.* Safety Alignment tasks test recognition of threat patterns (credential leaks, prompt injections, malicious skills), and each additional security-related entry—even amid noise—increases the probability of matching a relevant threat type. A single-run experiment suggests a sharp non-monotonic peak at five facts; the $N = 5$ replication does not confirm it, underscoring the need for multi-run evaluation on small task sets.

Table 12. Safety Alignment task completion on the live agent benchmark: seven-point memory size sweep plus typed policy prototype, $N = 5$ trials per condition (400 total runs), bootstrap 95% CI. Memory poverty (1–3 facts) reliably degrades below the no-memory baseline; richer threat pattern workspaces match or exceed it.

Facts	Condition	Mean ($N = 5$)	95% CI
0	No memory	0.318	[0.284, 0.352]
1	Severe poverty	0.270	[0.216, 0.324]
3	Over-consolidated (C4)	0.264	[0.208, 0.326]
5	Curated (oracle)	0.334	[0.242, 0.422]
10	Mild bloat	0.302	[0.260, 0.356]
25	Medium bloat	0.342	[0.276, 0.392]
55	**Large memory (C0)**	**0.390**	**[0.344, 0.442]**
5	**Typed-from-C0 (keyword)**	**0.380**	**[0.334, 0.418]**

8.3. A Typed Consolidation Prototype

The cumulative threat pattern effect suggests that the optimal eviction policy may be knowledge type-dependent. We prototype a typed consolidation policy for this one type, where a keyword-based session-type classifier labels each of the 55 entries in the large workspace as a threat pattern or generic, then evicts the 50 generic noise entries, retaining the 5 threat pattern facts. The resulting 5-fact workspace achieves 0.380 [0.334, 0.418], which is comparable to both the oracle 5-fact curated workspace (0.334, CI overlap) and the full 55-fact condition (0.390, CI overlap). The classifier autonomously recovers oracle-level retention quality from a large workspace *without oracle supervision,* supporting typed consolidation *for threat pattern memory in this setting.* We note that the setting is a relatively simple classification problem—five clearly security-related facts against fifty generic entries—and the classifier is evaluated only against benign generic noise; adversarially similar distractors that contain security vocabulary but are task-irrelevant remain for future work. We therefore treat this as preliminary evidence for the threat pattern type, not a validation of typed consolidation in general.

The classifier is a keyword matcher. Threat pattern keywords include injection, malicious behavior, exploits, credentials, leaks, exfiltration, webhooks, API keys, misinformation, destructive shell commands, hardcoded secrets, and passwords. All 5 relevant safety facts in the 55-fact workspace match, and all 50 noise entries ("browsed documentation page *i*") are unmatched and evicted. Per-task attribution is consistent with the action-dependent structure (Table 13): Task 4 (authority restriction) drops from 0.70 to near 0 under poverty across all five trials, while Task 9 (misinformation) is sensitive to the misinformation fact specifically. These fact-to-task associations—the kind LongMemEval-S lacks—are suggestive of specific facts gating specific tasks, though with $N = 5$ per condition on 10 tasks, we read them as indicative rather than definitive.

Table 13. Per-task attribution (Safety Alignment, mean over $N = 5$). Under poverty, tasks that appear to depend on a specific fact lose most of their score, consistent with specific facts gating specific tasks—an action-dependent structure absent from single-subject QA benchmarks. With $N = 5$ on 10 tasks, we read this as indicative.

Task	Required Fact Type	No Memory	Poverty (1–3 Facts)
Task 4 (authority restriction)	authority pattern	0.70	≈0.00
Task 9 (misinformation detection)	misinformation fact	varies	≈0.00

Policy specification.

To make the prototype reproducible, Table 14 gives the *proposed* type taxonomy—the keyword sets, the per-type retention rule, and an example—and Algorithm 3 states the consolidation procedure. Only the threat pattern branch is exercised by our experiment; the precision and relational branches are specified for completeness but not evaluated. The policy is deterministic and requires no supervision or LLM judge. Each entry is assigned the first matching type in priority order (threat pattern, then precision, then relational, else generic), and the per-type retention rule decides what survives consolidation. Threat pattern entries are retained broadly because their task relevance is high and their volume low; precision entries (paths, endpoints, deadlines) would be deduplicated to the most specific or most recent; relational entries (preferences, roles, state) would retain the most recent and evict superseded versions; generic entries are evicted. The threat pattern keyword list was fixed *before* inspecting the WildClawBench task instances—it derives from the Safety Alignment category description, not from the specific tasks—so the prototype is not tuned on task outcomes.

Algorithm 3: Typed consolidation (budget B)

Input: workspace entries E; keyword sets $K_{\text{threat}}, K_{\text{prec}}, K_{\text{rel}}$; budget B
Output: retained set R

$R \leftarrow \emptyset$;
foreach *entry* $e \in E$ **do**
  type$(e) \leftarrow$ first $t \in$ (threat, prec, rel) with $K_t \cap \text{tokens}(e) \neq \emptyset$, else GENERIC;
end
retain all THREAT-PATTERN entries into R;
add PRECISION entries to R, deduplicated to most-specific / newest;
add RELATIONAL entries to R, keeping most-recent, dropping superseded;
discard GENERIC entries;
if $|R| > B$ **then**
  keep the B highest-priority entries (threat > precision > relational);
end
return R;

Table 14. Proposed typed consolidation policy: type taxonomy, matching keywords/rules, retention rule, and example. Only the threat pattern branch is evaluated in this paper; precision and relational branches are specified but not yet tested. Types are matched in priority order (top to bottom); the first match wins.

Type	Keywords/Rule	Retention Policy	Example
Threat pattern	credential, injection, webhook, API key, exploit, leak, exfiltration, malicious, misinformation, authority	retain broadly	"hardcoded API key detected"
Precision	file path, endpoint, deadline, command, version	retain most specific/newest	"api.internal:8000"
Relational	preference, role, team, state update	retain recent, evict superseded	"user prefers dark mode"
Generic	no keyword match	evict	"browsed documentation page 12"

8.4. Toward Type-Dependent Retention

Our evidence establishes one type directly. **Threat pattern memory,** on the Safety Alignment tasks, is cumulatively beneficial; additional security-related entries help, poverty hurts, and a keyword classifier that retains threat pattern facts and evicts generic noise recovers near-oracle performance (Table 12). This is the one retention policy the experiments support.

The threat pattern result motivates—but does not by itself establish—a broader hypothesis: that the optimal retention strategy depends on the semantic *type* of the memory content, and that a single global eviction threshold is therefore the wrong abstraction. We state the two further types as *proposed, not yet evaluated:*

- **Precision memory** (file paths, API endpoints, deadlines) is conjectured to be sole-carrier dominant—it retains the single most precise relevant fact, evicts redundant or stale entries. The high-credit sole-carrier sessions of Section 6 are a suggestive retrieval signature, but we do not test a precision-specific retention policy here.
- **Relational memory** (user preferences, team structures) is conjectured to be recency-dominant—it retains recent state, evicts superseded facts. This too is a hypothesis; we do not evaluate it under its proposed policy.

Evaluating precision and relational retention under their own policies, and on more challenging distractors than benign generic noise, is the natural next step. The evidence here supports the narrower claim that *at least one* memory type (threat pattern) benefits from type-aware retention over a global threshold. Whether the full three-type taxonomy holds is left to future work.

9. Discussion and Conclusions

9.1. One Causal Arc

The three studies are not independent results. The retrieval-limited regime (Section 4) shows that better generation and better weights cannot help when linear retrieval cannot surface cross-session evidence. This diagnosis sharpens in the injection study (Section 5): the right session is usually found, but the right evidence is not injected; whole-session injection closes much of the gap, with the structure-specific benefit concentrated where ordering matters. The injection study's drop-one attribution (Section 6) produces session credit labels that are precisely the training signal the consolidation study (Sections 7 and 8) needs. The unifying thread is pattern recognition over interaction traces: recognizing which facts match a query, which session patterns carry evidence, and which knowledge types to retain.

9.2. Pattern Recognition View

Each stage is a recognition problem with a different unit: retrieval recognizes query–fact matches; credit attribution recognizes evidence-bearing session patterns via a causal drop-one test; typed consolidation recognizes semantic knowledge types and matches a retention policy to each. The empirical payoff of this view is concrete: a trivial keyword recognizer recovers near-oracle consolidation, and the same credit signal that explains multi-session difficulty (a 13% positive-credit rate) also supplies the labels for retention.

9.3. Limitations

- **Single retrieval benchmark.** The retrieval and injection studies use one benchmark whose single-subject structure limits the benefit of dense retrieval and prevents learning of consolidation policies; multi-entity benchmarks may behave differently.
- **Consolidation scope.** The consolidation study evaluates the Safety Alignment category only, with synthetic curated workspaces, and the typed classifier is tested against benign noise rather than adversarially similar distractors.
- **Prototype, not learned policy.** We prototype, but do not yet learn end-to-end, a typed consolidation policy with per-type thresholds via online RL on terminal task rewards. This is the immediate next step.
- **Statistical scope.** Claims rest on Wilson and bootstrap confidence intervals rather than multi-seed hypothesis tests.

https://doi.org/10.3390/a19070607

- **Context budget instrumentation.** The largest WildClawBench workspace fits within the context limit, so it measures contextual priming rather than retrieval index failure. A top-k retrieval variant under explicit token budgets is future work.

9.4. Conclusions

We presented MEMTIER, a tiered memory architecture and consolidation framework for long-running LLM agents, and three related diagnostic studies. A tiered episodic–semantic memory reaches 0.382/0.412 on LongMemEval-S with a 7B model on a 6 GB GPU, but a corrected single-pipeline ablation and oracle decomposition place the system in a retrieval-limited regime, where the gains come from semantic pre-population and two-stage scoping, not from the multi-signal retrieval scaffold, which is an operational pass-through of BM25 top-k in this setting. Session-level injection improves multi-session (+0.120) and knowledge update (+0.205) accuracy, with a controlled comparison isolating the part due to session structure and a drop-one evidence-coverage pass producing session credit labels. Casting consolidation as an MDP reveals three structural failure modes on single-subject benchmarks, while, on a live agent benchmark, a keyword-based typed policy recovers near-oracle performance for one memory type—threat pattern memory—without supervision. We read this as preliminary evidence that the semantic type of a memory, not merely its volume, can determine how it should be retained. Establishing this beyond threat pattern memory and closing the attribution-to-consolidation loop end-to-end are the main directions we leave open. The unifying theme is that deciding what a long-running agent should remember is best understood as recognizing which interaction patterns carry evidence.

Author Contributions: Conceptualization, B.S. and L.R.; methodology, B.S.; software, B.S.; validation, B.S.; investigation, B.S.; writing—original draft preparation, B.S.; writing—review and editing, B.S. and L.R.; supervision, L.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The benchmarks used are publicly available. The MEMTIER plugin is released at https://github.com/sidikbro/openclaw-memtier (accessed on 20 July 2026). Analysis, evaluation, and table generation scripts are being added to the same repository.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing reasoning and acting in language models. *arXiv* **2023**, arXiv:2210.03629. [CrossRef]
2. Packer, C.; Wooders, S.; Lin, K.; Fang, V.; Patil, S.G.; Stoica, I.; Gonzalez, J.E. MemGPT: Towards LLMs as operating systems. *arXiv* **2023**, arXiv:2310.08560. [CrossRef]
3. OpenClaw Team. OpenClaw. 2026. Available online: https://github.com/openclaw/openclaw (accessed on 20 July 2026).
4. Sun, H.; Zeng, S.; Zhang, B. H-MEM: Hierarchical memory for high-efficiency long-term reasoning in LLM agents. In Proceedings of the European Chapter of the Association for Computational Linguistics (EACL), Rabat, Morocco, 24–29 March 2026; pp. 341–350. [CrossRef]
5. Liu, J.; Su, Y.; Xia, P.; Han, S.; Zheng, Z.; Xie, C.; Ding, M.; Yao, H. SimpleMem: Efficient lifelong memory for LLM agents. *arXiv* **2026**, arXiv:2601.02553. [CrossRef]
6. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv* **2020**, arXiv:2005.11401. [CrossRef]
7. Xu, W.; Liang, Z.; Mei, K.; Gao, H.; Tan, J.; Zhang, Y. A-MEM: Agentic memory for LLM agents. *arXiv* **2025**, arXiv:2502.12110. [CrossRef]
8. Yan, S.; Yang, X.; Huang, Z.; Nie, E.; Ding, Z.; Li, Z.; Ma, X.; Schütze, H.; Tresp, V.; Ma, Y. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv* **2025**, arXiv:2508.19828. [CrossRef]

9. Maharana, A.; Lee, D.; Tulyakov, S.; Bansal, M.; Barbieri, F.; Fang, Y. Evaluating very long-term conversational memory of LLM agents. *arXiv* **2024**, arXiv:2402.17753. [CrossRef]
10. Wu, D.; Wang, H.; Yu, W.; Zhang, Y.; Chang, K.; Yu, D. LongMemEval: Benchmarking chat assistants on long-term interactive memory. *arXiv* **2025**, arXiv:2410.10813. [CrossRef]
11. Ding, S.; Dai, X.; Xing, L.; Ding, S.; Liu, Z.; Yang, J.; Yang, P.; Zhang, Z.; Wei, X.; Fang, X.; et al. WildClawBench: A benchmark for real-world, long-horizon agent evaluation. *arXiv* **2026**, arXiv:2605.10912. [CrossRef]
12. He, Z.; Wang, Y.; Zhi, C.; Hu, Y.; Chen, T.-P.; Yin, L.; Chen, Z.; Wu, T.A.; Ouyang, S.; Wang, Z.; et al. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks. *arXiv* **2026**, arXiv:2602.16313. [CrossRef]
13. Uddin, M.N.; Shubham, K.; Blanco, E.; Baral, C.; Wang, G. From recall to forgetting: Benchmarking long-term memory for personalized agents. *arXiv* **2026**, arXiv:2604.20006. [CrossRef]
14. Hu, Y.; Wang, Y.; McAuley, J. Evaluating memory in LLM agents via incremental multi-turn interactions. *arXiv* **2025**, arXiv:2507.05257. [CrossRef]
15. Du, P. Memory for autonomous LLM agents: Mechanisms, evaluation, and emerging frontiers. *arXiv* **2026**, arXiv:2603.07670. [CrossRef]
16. Zhu, J.; Li, J.; Zhang, C.; Liu, J.; Yang, M. HeLa-Mem: Hebbian Learning and Associative Memory for LLM Agents. *arXiv* **2026**, arXiv:2604.16839. [CrossRef]
17. Strich, J.; Isgorur, E.K.; Trescher, M.; Biemann, C.; Semmann, M. T2-RAGBench: Text-and-Table Benchmark for Evaluating Retrieval-Augmented Generation. In Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics, Rabat, Morocco, 24–29 March 2026; Volume 1: Long Papers, pp. 165–191.
18. Tavakoli, M.; Salemi, A.; Ye, C.; Abdalla, M.; Zamani, H.; Mitchell, J.R. Beyond a million tokens: Benchmarking and enhancing long-term memory in LLMs. *arXiv* **2025**, arXiv:2510.27246. [CrossRef]
19. Chhikara, P.; Khant, D.; Aryan, S.; Singh, T.; Yadav, D. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv* **2025**, arXiv:2504.19413. [CrossRef]
20. Kang, J.; Ji, M.; Zhao, Z.; Bai, T. Memory OS of AI agent. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Suzhou, China, 4–9 November 2025; pp. 25961–25970. [CrossRef]
21. Zhao, Y.; Yuan, B.; Huang, J.; Yuan, H.; Yu, Z.; Xu, H.; Hu, L.; Shankarampeta, A.; Huang, Z.; Ni, W.; et al. AMA-Bench: Evaluating long-horizon memory for agentic applications. *arXiv* **2026**, arXiv:2602.22769. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.