Persistent Memory Infrastructure for Autonomous and Cooperative AI Systems

AltLab Research Paper · ALT-RP-002

Safari — Founder & Research Lead, AltLab

June 2026

1 Abstract

Modern AI systems built on large language models demonstrate strong reasoning ability within a single interaction but lack durable, well-organized memory across interactions. Three deficiencies recur across current approaches: information that matters is lost once it leaves the active context; systems that do retain information accumulate it without bound, degrading retrieval quality through noise; and no widely deployed system gives multiple cooperating agents an efficient, consistent way to share what they have learned. This paper proposes **AltMemory**, a persistent memory infrastructure for autonomous and cooperative AI systems. AltMemory organizes agent memory into working, episodic, and semantic stores; introduces a Memory Ranking Engine and an Intelligent Forgetting Engine as explicit, first-class subsystems; and extends the architecture with Shared Memory Spaces for multi-agent cooperation. We describe the architecture, situate it against existing memory mechanisms (ChatGPT Memory, Claude Memory, MemGPT, LangChain Memory, and AutoGen), and propose a concrete experimental program covering semantic retrieval, forgetting, and shared-memory cooperation. We are explicit about the architecture’s current limitations: AltMemory is proposed and partially implemented (AltMemory v0.1), not yet validated at the

scale or generality this paper describes.

---

2 1. Introduction

Large language models have become capable enough to act as autonomous agents — planning multi-step tasks, calling tools, and operating with limited supervision. What they have not become is *persistent*. By default, an agent's working state is whatever fits in its context window for the current call; once that call ends, nothing is retained unless an external system explicitly captures and re-injects it. This is adequate for short, single-purpose interactions. It is not adequate for an agent expected to operate over days or weeks, recall what a specific user prefers, avoid repeating a mistake it already made, or coordinate with other agents working on a shared task.

The need for persistent memory has produced several partial solutions, surveyed in Section 4: per-product memory features layered onto chat assistants, retrieval-augmented generation pipelines, and research systems that simulate larger context through paging. Each addresses part of the problem. None, to our knowledge, treats memory *ranking* and memory *forgetting* as architectural primitives with the same standing as storage and retrieval, and none provides a native mechanism for multiple agents to share a common memory space with conflict-aware coordination.

AltMemory is AltLab's proposed answer to this gap. It is the architectural successor to the conceptual design introduced in AltLab Paper #001 and the engineering basis for the AltMemory v0.1 implementation, extended here with an explicit cooperative multi-agent layer. This paper describes the full architecture, the problems it targets, the experiments needed to validate it, and the limitations we are aware of at this stage.

---

3 2. Problem Statement

We organize the memory problem into four concrete failure modes observed in current AI systems.

3.1 2.1 AI Forgets Important Information

Without persistent storage, anything outside the active context window is unavailable to the model, regardless of how important it was. A user's stated preference, a hard-won correction to an earlier mistake, or a critical fact established early in a long task can all be silently dropped the moment they scroll out of context. This is not selective forgetting; it is undifferentiated loss, equally likely to discard a trivial remark and a load-bearing fact.

3.2 2.2 Unlimited Memory Causes Noise

The naive fix — store everything, forget nothing — creates a different failure. As a memory store grows without bound, retrieval at query time must sift through an increasing volume of low-value, redundant, or outdated material. Past a certain size, an unranked, unpruned memory store becomes actively harmful to retrieval quality: the system is more likely to surface something merely topically similar than something actually useful, simply because there is more of the former.

3.3 2.3 AI Cannot Prioritize Memories Effectively

Most retrieval mechanisms in production today rank candidates by embedding similarity alone. Similarity is a measure of topical relatedness, not of importance, reliability, or recency. A memory that is semantically close to a query but was a one-off, low-confidence observation can outrank a memory that is more central to the user's actual needs. Without a prioritization signal independent of similarity, systems cannot distinguish a passing remark from a standing fact.

3.4 2.4 Multiple Agents Cannot Share Memories Efficiently

As soon as more than one agent is involved in a task, new problems appear that single-agent memory designs do not address: which agent's observation takes precedence when two disagree; which memories are private to one agent versus visible to all; how a shared memory store stays consistent when multiple agents write to it concurrently; and how an agent knows that a piece of shared context has changed since it last read it. Conventional memory designs, built around a single reader and writer, have no native answer to any of these.

---

4 3. Related Work

We compare AltMemory against five existing approaches to AI memory. This comparison is based on publicly described behavior of these systems; we have not benchmarked them directly, and vendor implementations change over time.

System	Core Mechanism	Memory Typing	Ranking Beyond Similarity	Forgetting Mechanism	Multi-Agent Sharing
ChatGPT Memory	Persisted user facts/preferences surfaced across sessions	Largely undifferentiated (flat fact store)	Limited; mostly recency and explicit user edits	User-driven (manual deletion); limited automatic decay	Not designed for multi-agent use
Claude Memory	Persisted, user-controlled memory of past conversations	Largely undifferentiated	Limited; primarily relevance to current conversation	User-driven (clearable); no published automatic decay model	Not designed for multi-agent use
MemGPT	OS-inspired paging of context in and out of an LLM's active window	Working vs. archival distinction (paging)	Heuristic, paging-driven rather than multi-signal ranking	Implicit via paging policy, not an explicit decay/importance model	Single-agent design; no native sharing layer
LangChain Memory	Library of memory abstractions (buffer, summary, vector-store-backed) over a single conversation	Developer-defined per memory class used	Whatever the underlying retriever provides (typically similarity)	Developer-implemented; not built in	Not a native feature; would require custom integration
AutoGen	Multi-agent conversation framework with shared conversational state	Conversation-scoped context, not a persistent long-term store by default	Not a core focus of the framework	Not a core focus of the framework	Strong multi-agent *conversation* support, but not long-term shared *memory* with ranking/forgetting

Summary of the gap. ChatGPT Memory and Claude Memory demonstrate that persistent, cross-session memory is valuable to end users, but both are largely flat stores without explicit ranking or decay models, and neither is designed for multi-agent use. MemGPT's paging concept is a meaningful step toward managing memory under a context budget, but its memory management is driven by paging policy rather than an explicit, tunable importance/forgetting model. LangChain Memory provides useful primitives but leaves ranking, forgetting, and sharing as integration work for the developer. AutoGen solves multi-agent *conversation*, which is a different problem from multi-agent *persistent memory*: it does not, by itself, give agents a durable, ranked, shared store that outlives a conversation.

AltMemory's contribution relative to this landscape is to make ranking, forgetting, and multi-agent sharing explicit, first-class subsystems of the architecture rather than implementation details left to whoever integrates the system.

---

5 4. AltMemory Architecture

AltMemory is composed of eight interacting components, shown in Figure 1.

5.1 4.1 Working Memory

Working memory is the agent's active context for the current task — the material actually passed to the model. It functions as a cache rather than a store of record: information only persists beyond the current task if it is explicitly written into one of the long-term stores below.

5.2 4.2 Episodic Memory

Episodic memory stores time-ordered records of specific events: completed tasks, stated preferences at a point in time, corrected errors. It preserves *sequence and context*, which is necessary for tasks that depend on understanding what happened, in what order, and why — information that a flat fact store discards.

5.3 4.3 Semantic Memory

Semantic memory stores generalized, time-independent facts and concepts distilled from episodes or stated directly: stable user preferences, durable facts about the environment, and conclusions an agent has reached after repeated observation. Where episodic memory answers "what happened," semantic memory answers "what is true."

5.4 4.4 Memory Ranking Engine

The Memory Ranking Engine scores candidate memories both at write time (to assign initial importance) and at retrieval time (to order results for a given query). Ranking combines multiple signals — similarity, recency, importance, and access frequency — rather than relying on similarity alone, directly addressing the prioritization failure described in Section 2.3.

5.5 4.5 Intelligent Forgetting Engine

The Intelligent Forgetting Engine is the architecture's answer to unbounded memory growth (Section 2.2). It runs as a periodic process that scores stored memories for continued relevance and applies decay, archiving, or deletion as appropriate. It is described in detail in Section 5.

5.6 4.6 Shared Memory Spaces

Shared Memory Spaces are memory partitions explicitly scoped for access by more than one agent, carrying provenance metadata (which agent wrote a given memory), access scope (which agents may read or write), and conflict-resolution state for contradictory entries. This is the architecture's direct response to the multi-agent sharing failure described in Section 2.4, and is discussed further in Section 6.

5.7 4.7 Vector Database

The vector database stores embeddings for episodic, semantic, and shared memories and supports approximate nearest-neighbor search. AltMemory treats this layer as infrastructure rather than a site of architectural novelty: existing vector database technology (e.g., pgvector) is adequate for this role, and the architecture's contribution lies in the layers above it.

5.8 4.8 Retrieval Engine

The Retrieval Engine accepts the agent's current context as a query, performs similarity search against the vector database, and applies the Memory Ranking Engine's scoring before returning a bounded, ordered set of memories. It is the component that converts "what is topically related" into "what should actually be surfaced now."

5.9 4.9 Embedding Engine and Response Generator

The Embedding Engine converts memory content and queries into vector representations (OpenAI embeddings in the current implementation; BGE and Jina are tracked as future provider options). The Response Generator is the underlying model call that consumes retrieved memories alongside working memory and produces the agent's output, while also proposing new candidate memories for the Memory Ranking Engine to evaluate.

Figure 1 (see accompanying architecture diagram) shows the full data flow: User → AI Agent → Memory Manager (Working / Episodic / Semantic / Shared Memory / Ranking / Forgetting) → Embedding Engine → Vector Database → Retrieval Engine → Response Generator, with the generated response flowing back to the agent and user.

---

6 5. Intelligent Forgetting

Forgetting is treated in AltMemory as a designed process, not an accident of storage limits. It rests on four mechanisms.

6.1 5.1 Memory Decay

Each memory carries a decay score that decreases over time when the memory is not accessed, and is refreshed (wholly or partially) when the memory is retrieved and used. Decay is a continuous signal, not a binary state — a memory's priority in ranking falls gradually as it ages and goes unused, rather than disappearing abruptly.

6.2 5.2 Importance Scoring

Independent of decay, each memory carries an importance score reflecting how consequential it is judged to be, set at write time (explicitly, or inferred from context) and revisable over time. Importance acts as a counterweight to decay: a highly important memory that has not been accessed recently should decay more slowly than a low-importance one.

6.3 5.3 Automatic Forgetting

The Forgetting Engine runs on a schedule, recomputing decay scores and applying threshold-based actions without requiring manual intervention. This is what allows the memory store to remain usable as it grows, directly addressing the noise problem from Section 2.2: a store that forgets well should not degrade in retrieval quality merely because it has existed longer.

6.4 5.4 Archiving

Forgetting is not necessarily deletion. AltMemory distinguishes three possible actions on a low-priority memory: **decay** (lower its ranking weight but keep it fully active), **archive** (remove it from default retrieval while retaining it in cold storage), and **delete** (permanent removal). Archiving exists specifically because the cost of permanently losing a memory that later proves relevant is typically much higher than the cost of temporarily excluding it from default retrieval; AltMemory's default policy favors archiving over deletion unless deletion is explicitly enabled.

---

7 6. Cooperative AI Memory

Cooperative use is the primary extension this paper makes beyond AltLab Paper #001's single-agent design.

7.1 6.1 How Multiple Agents Share Memories

Agents write to Shared Memory Spaces using the same episodic/semantic structure as private memory, with additional provenance metadata recording which agent produced a given entry and when. Read access can be scoped per agent or per task, so that not all of an agent's private reasoning is exposed to its collaborators by default — only what is explicitly written to a shared space.

7.2 6.2 How Agents Access Common Context

Because Shared Memory Spaces are indexed in the same vector database as private memory, an agent's Retrieval Engine query can span both its own private memories and any shared spaces it has access to in a single retrieval pass, with the Memory Ranking Engine applying the same multi-signal scoring across both. This avoids requiring agents to run separate retrieval passes against separate stores and then merge results heuristically.

7.3 6.3 How Agents Coordinate Decisions

When two agents write contradictory information to a shared space, AltMemory does not assume a single universal resolution rule is correct for every task. We propose that conflict resolution be configurable per shared space along three policies — **last-write-wins** (simplest, appropriate for low-stakes, fast-changing shared state), **confidence-weighted merge** (the entry with higher source confidence or corroboration takes priority), and **explicit adjudication** (conflicting entries are flagged and require a designated agent or human to resolve before either is treated as authoritative). The choice of policy is a task-level decision, not an architectural constant, because the cost of being wrong differs sharply across use cases.

We treat this layer as the least mature part of the architecture. The conflict-resolution policies above are proposed designs; Experiment #3 (Section 7.3) is intended to evaluate them rather than assume any one is correct.

---

8 7. Experiments

We propose three experiments. None have been run to completion at the scale described here; results from a smaller-scale validation on AltMemory v0.1 will be reported in a follow-up paper.

8.1 7.1 Experiment #1 — Semantic Memory Retrieval

Goal. Measure whether multi-signal ranking (similarity + recency + importance) retrieves more useful semantic memories than similarity-only retrieval.

Design. Construct a benchmark of queries against a seeded semantic memory store with human-annotated relevance labels. Compare similarity-only retrieval against the full Memory Ranking Engine, holding the underlying vector search and embedding model constant across conditions.

Metrics. Recall and precision against the relevance labels; retrieval latency, to confirm ranking does not materially increase end-to-end response time.

8.2 7.2 Experiment #2 — Intelligent Forgetting Benchmark

Goal. Measure whether the decay/archive policy described in Section 5 maintains or improves retrieval quality over time relative to unbounded accumulation.

Design. Run extended simulated sessions (thousands of memory writes) under three conditions: no forgetting, decay-only, and decay-with-archiving. Periodically evaluate retrieval quality against a fixed query set as the store grows.

Metrics. Recall and precision over time (to detect degradation under each policy); a context-retention metric measuring whether task-critical facts written early remain correctly retrievable later in the session; retrieval latency as store size grows.

8.3 7.3 Experiment #3 — Shared Memory Benchmark

Goal. Evaluate whether Shared Memory Spaces improve multi-agent task outcomes, and measure the practical cost of each conflict-resolution policy from Section 6.3.

Design. Construct a cooperative task requiring two or more agents to track and act on shared state (e.g., a jointly maintained project status). Compare task performance with no sharing, naive shared storage without conflict resolution, and each of the three proposed conflict-resolution policies.

Metrics. Recall accuracy of shared facts across agents; precision, specifically the rate at which an agent retrieves an incorrect or superseded shared memory; write-to-read propagation latency between agents; task-level success rate as a downstream outcome measure.

---

9 8. Limitations

We state these directly rather than deferring them to future work, since they bear on how the architecture described here should currently be interpreted.

- **Validation status.** This paper describes a proposed architecture. Only a subset (the single-agent save/retrieve/forget loop) has a working implementation at the time of writing (AltMemory v0.1); Shared Memory Spaces and the full Memory Ranking Engine as described here are design proposals pending implementation and the experiments in Section 7.
- **Ranking weights are not learned.** The current ranking design combines signals via configured weights rather than a learned, task-adaptive function. This is deliberate for interpretability at this stage, but it means ranking quality depends on manual tuning rather than automatic adaptation.
- **Conflict resolution is unvalidated.** The three policies proposed in Section 6.3 are reasonable candidates, not empirically validated choices. We do not yet know which performs best under which conditions, or whether a single task should be allowed to mix policies across different shared spaces.
- **Single embedding provider.** The architecture currently depends on OpenAI’s embeddings API. While the embedding layer is abstracted to support future providers (BGE, Jina), no comparative evaluation across providers has been performed.
- **No formal multi-tenant security model.** Shared Memory Spaces assume cooperating, non-adversarial agents. The architecture does not yet address what happens when one agent in a shared space is compromised or behaves adversarially toward the others.
- **Forgetting risk is asymmetric and unresolved.** Defaulting to archive-over-delete reduces but does not eliminate the risk of losing relevant information; we have not yet established a reliable way to detect, after the fact, that an archived memory should have remained active.

---

10 9. Future Work

- **Hierarchical memory**, organizing episodic traces into summarized higher-level structures so retrieval can operate at a level of abstraction appropriate to the query, rather than always searching the finest-grained store.
- **Memory compression**, to reduce the storage and retrieval cost of large episodic histories without discarding their essential content.
- **Learned ranking and forgetting**, replacing the current hand-tuned weighting with policies adapted from observed usage patterns, once enough usage data exists to do so reliably.
- **Adversarial robustness for Shared Memory Spaces**, addressing the security gap noted in Section 8.
- **Cross-provider embedding evaluation**, benchmarking OpenAI, BGE, and Jina embeddings on the retrieval tasks

proposed in Section 7.

---

11 10. Conclusion

Persistent memory remains one of the central unsolved problems for autonomous AI agents, and the four failure modes in Section 2 — undifferentiated forgetting, unbounded noise, poor prioritization, and inefficient multi-agent sharing — are each visible in some form across the existing systems surveyed in Section 4. AltMemory proposes an architecture that treats ranking, forgetting, and cooperative sharing as explicit, designed subsystems rather than integration afterthoughts. We have tried throughout this paper to be precise about what is implemented, what is proposed, and what remains genuinely open: the single-agent core of AltMemory exists and runs; the cooperative layer and the multi-signal ranking and forgetting policies described here are proposals awaiting the experiments outlined in Section 7. We expect the hardest open problems to remain in multi-agent conflict resolution and in learned, rather than hand-tuned, ranking and forgetting — both of which involve tradeoffs that are unlikely to have one correct answer independent of the task. We intend to report empirical results from this experimental program in a subsequent AltLab research paper.

---

12 References

1. AltLab Research Division. *Persistent Memory Systems for Autonomous AI Agents.* AltLab Research Paper #001, 2026.
2. Packer, C. et al. *MemGPT: Towards LLMs as Operating Systems.* arXiv preprint, 2023.
3. LangChain. *LangChain Memory Documentation.* langchain.com, accessed 2026.
4. Microsoft Research. *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation.* arXiv preprint, 2023.
5. OpenAI. *Memory and New Controls for ChatGPT.* openai.com, 2024.
6. Anthropic. *Claude's Memory Feature Documentation.* anthropic.com, accessed 2026.
7. Lewis, P. et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* NeurIPS, 2020.

AltLab Research Paper ALT-RP-002. AltLab — Persistent Memory Infrastructure for Autonomous and Cooperative AI Systems. June 2026.