Cortex: A Governed Long-Term Memory Engine for LLM Agents

Subomi Olagoke *Preprint (draft), 2026. Correspondence: solagoke21@gmail.com*

Abstract

Long-term memory has become standard equipment for LLM agents. Systems such as MemGPT/Letta, Generative Agents, MemoryBank, A-MEM, and Mem0 let an agent carry information across sessions. Almost all of this work targets capacity and recall: fitting more history into a finite context window and retrieving the right fact at scale. A younger line of 2026 work argues that the harder problem is not recall but governance. That means deciding what should be written, reconciling contradictions, scoping memory across users and tenants, auditing what the agent knows and why, and forgetting under policy. So far that literature is mostly conceptual frameworks and single-property benchmarks, with very few running systems to point at.

This paper presents Cortex, a transparent, self-pruning memory engine, available as both a REST API and an MCP server, that implements five governance behaviors in one system: consolidation by merging restatements, salience-scaled adaptive forgetting, contradiction supersession, per-namespace tenant isolation, and an exportable audit trail. We describe the design and evaluate the governance behaviors directly, using real local sentence embeddings. Consolidation shrinks the memory footprint by 89% on our workload without dropping any important fact. Deterministic contradiction reconciliation reaches 100% precision but only 40% recall, which is safe but conservative, and quantifies exactly why an LLM classification step is needed; it also gives the field's deterministic-versus-LLM debate a concrete number. Cross-tenant isolation holds under a memory-extraction attack, with zero leaks across 50 adversarial retrieval slots, because isolation is structural rather than filtered. The audit trail records provenance, access, and deletion, and exports on demand. We position Cortex against the emerging governance literature, argue that the missing artifact in that literature is a running, auditable system, and set out the standard-benchmark experiments still needed to turn these results into comparative claims. Throughout, we are explicit about which results are established and which remain to be run.

1. Introduction

An LLM agent with no memory starts from zero every session. The usual fix is to embed prior conversation into a vector store and retrieve the top matches into the prompt. That approach has carried the field a long way, and a rich ecosystem of memory architectures now exists [MemGPT; GenerativeAgents; MemoryBank; A-MEM; Mem0]. Nearly all of it shares one framing: memory exists to get past a finite context window and to persist facts across sessions, and progress is measured by how much can be stored and how accurately it can be recalled.

That framing has a blind spot. As agents become persistent, multi-user, and long-lived, the pressing questions stop being "can it recall the fact?" and become questions of governance:

- **Validation.** Should this be written to memory at all, and is it true?
- **Reconciliation.** When new information contradicts an old memory, which one wins, and is the resolution correct?
- **Scoping.** Whose memory is this? Can one user's memory leak into another's, or across tenants?
- **Auditability.** Can we answer what the agent knows about a user, why it knows it, who accessed it, and what was deleted?
- **Forgetting.** Can memory be removed under policy, whether for privacy, staleness, or safety, with a guarantee and a record?

A wave of 2026 work has begun to name exactly this gap. Governance frameworks such as SSGM [SSGM] break agent memory into consistency verification, temporal decay, and access control before consolidation. Security surveys frame a full memory lifecycle crossed with confidentiality, integrity, availability, and governance, and coin the term "mnemonic sovereignty" [MnemonicSovereignty]. A 435-paper survey of persistent agents puts it bluntly: the field "concentrates more heavily on accumulating and retrieving state than on governing, recovering, or relinquishing it" [AlwaysOnAgents]. New benchmarks probe individual governance properties, including contradiction handling [MemConflict], memory staleness [STALE], and belief revision [BeliefShift], and they consistently find current systems weak. Profile-centric memory systems have been reported to reconcile single-hop conflicts as poorly as roughly 18% [MemConflict], and even the strongest models detect silently invalidated memories only about 55% of the time [STALE].

Two things stand out about this literature. It is mostly conceptual, offering frameworks, taxonomies, and threat models with very few running systems. And its evaluations are fragmented: each benchmark isolates a single property, and none connects back to the recall benchmarks the field already uses. A system can therefore top a recall leaderboard while being trivially poisoned [MINJA], leaking across users [MEXTRA], and unable to reconcile contradictions [MemConflict].

Cortex is a system built into that gap. It is a memory engine, reachable through a REST API and an MCP server, whose design goal is not maximal recall but governed memory: memory you can see, audit, scope, and prune. Concretely, Cortex provides five behaviors:

1. **Consolidation.** Restating a known fact merges into the existing memory instead of duplicating it, so the store does not bloat.
2. **Adaptive forgetting.** Every memory has a salience-scaled forgetting curve. Unimportant memories decay below a threshold and are dropped, important ones resist forgetting, and recall reinforces them.
3. **Contradiction supersession.** New information that conflicts with an existing memory replaces it, and the replacement is recorded.
4. **Per-namespace scoping.** Memory is isolated per user (namespace) and shared across agents within a namespace, which gives multi-tenant isolation alongside deliberate cross-agent reuse.
5. **Auditability.** Every lifecycle event (create, merge, access, supersede, forget) is written to a per-memory trail and to an exportable engine-wide audit log.

Our contribution is a system together with a direct evaluation of its governance behaviors. It is not a new framing, since that framing now exists, and it is not a new benchmark. We report measured results for consolidation, deterministic reconciliation, isolation under attack, retrieval, and auditability. We are also clear about the limits: our current numbers come from a harness that uses real local sentence embeddings on synthetic workloads, and Section 5 says what that establishes while Sections 6 and 7 say what standard-benchmark evaluation still remains.

---

2. Related Work

2.1 Long-term memory architectures and the memory lifecycle

A useful way to compare memory systems is by the lifecycle of operations they support: write (encode), consolidate (summarize or abstract), retrieve, update (edit), and forget (evict). CoALA [CoALA] supplies the standard vocabulary, carrying the working, episodic, semantic, and procedural taxonomy of classical cognitive architectures (Soar, ACT-R) over to language agents.

Foundational systems sit at different points in this space. MemGPT/Letta [MemGPT] treats context like an operating system, paging memory between a finite main context and external stores and letting the agent edit its own core-memory blocks. Generative Agents [GenerativeAgents] keep an append-only natural-language memory stream plus reflections, retrieved by a weighted combination of recency, importance, and relevance. MemoryBank [MemoryBank] stands out as an early system with principled forgetting, using an Ebbinghaus forgetting curve to govern retention and reinforcement. Reflexion [Reflexion] stores verbal self-reflections as an episodic learning substrate. Voyager [Voyager] accumulates a procedural library of verified code, one of the few systems with a validation-before-write gate, though it validates that code runs rather than that a fact is true. HippoRAG [HippoRAG] builds a knowledge-graph index queried with Personalized PageRank. A-MEM [A-MEM] links memories into a Zettelkasten-style network whose notes can be revised after the fact. Mem0 [Mem0] is the most production-oriented, with an explicit ADD/UPDATE/DELETE/NOOP operator that an LLM applies to each extracted fact.

Across these systems, write and retrieve are universal, consolidate is common, update is uneven, and forget is rare. Where update or forget do appear, the decision is handed either to an unaudited LLM call (Mem0, A-MEM) or to a utility heuristic (MemoryBank decay), never to a governed, logged, policy-driven operation. Surveys of the space [MemorySurvey] confirm this distribution.

2.2 The emerging governance literature (2026)

Starting in early 2026, a separate line of work reframes memory as a governed resource. SSGM [SSGM] proposes an architecture that intercepts memory evolution with consistency verification, temporal-decay modeling, and access control before consolidation. A security survey organizes a six-phase memory lifecycle against integrity, confidentiality, availability, and governance, and argues that security "must be anchored in storage-time provenance, versioning, and policy-aware retention" [MnemonicSovereignty]. The Always-On Agents survey [AlwaysOnAgents] reviews 435 works and proposes AOEP-v0, a pilot

evaluation contract that scores state-mutation and recovery obligations rather than answer quality. It is the closest thing to a governed-memory evaluation so far, and its authors call it preliminary.

Cortex differs from this literature on one axis. It is a running, measurable system rather than a framework. Where SSGM specifies governance components in the abstract, Cortex implements a working subset (consolidation, decay, supersession, scoping, audit) and exposes it through an API and MCP. Where AOEP-v0 proposes an evaluation contract, Cortex is a candidate system to run against it.

2.3 Governance-adjacent problems and evaluations

Individual governance properties have their own, mostly 2026, literatures that do not yet compose.

On contradiction and belief revision, model-weight editing (ROME [ROME], MEMIT [MEMIT]) changes stored beliefs at the parameter level, which is distinct from reconciling entries in an external store. In the memory-store setting, MemConflict [MemConflict] benchmarks dynamic, static, and conditional conflicts and finds answer correctness diverging from retrieval. STALE [STALE] finds agents poor at noticing when a memory has silently gone stale. BeliefShift [BeliefShift] scores temporal belief consistency and evidence-driven revision. A methodological disagreement runs through this work, between LLM-driven reconciliation (as in Mem0) and deterministic supersession.

On memory poisoning and privacy, the attacks are strong and appear at top venues. AgentPoison [AgentPoison] backdoors agents through poisoned memory, reaching over 80% success at under 0.1% poison rate. MINJA [MINJA] injects malicious memories using ordinary user queries alone, at roughly 98% injection success. PoisonedRAG [PoisonedRAG] corrupts retrieval knowledge bases. On privacy, MEXTRA [MEXTRA] extracts other users' records from a shared memory module, which motivates user-level and session-level isolation. Defenses trail the attacks: certified defenses exist for a static corpus [RobustRAG], but the runtime, mutable-memory case has only early proposals [SMSR].

On recall, LongMemEval [LongMemEval], LoCoMo [LoCoMo], and DialSim [DialSim] measure recall, multi-session consistency, and temporal reasoning at scale. None of them scores auditability, scoping isolation, verifiable forgetting, or poisoning resistance.

The picture is a set of disconnected specialist tracks. Cortex is built so that a single system spans several of them, which is what makes a unified evaluation possible.

---

3. Design Goals

Cortex is organized around four goals that follow from the governance framing.

- **G1, Leanness.** Memory should not grow without bound. Restatements should consolidate, and low-value memories should decay out, so that context footprint tracks information rather than the length of the interaction.
- **G2, Consistency.** New information that contradicts an existing memory should replace it rather than pile up beside it, so the store does not hold mutually contradictory beliefs.

- **G3, Isolation with intentional sharing.** Memory should be isolated per user or tenant by default, while allowing several agents that serve the same user to share it on purpose.
- **G4, Transparency.** Every change to memory should be observable and exportable: what was written, by which agent, when it was accessed, and when it was superseded or forgotten.

G4 is the property the 2026 surveys single out as most under-served, and it is Cortex's main point of difference. G1 through G3 are shared in part with prior systems, but they are rarely all present at once and rarely exposed as first-class, inspectable operations.

---

4. System Design

4.1 Overview and data model

Cortex is a Python engine reachable two ways, through a multi-tenant REST API and through an MCP server, that share a single core so every caller gets identical behavior:

```
MCP client  ─┐
             ├─►  registry (per-namespace engine lookup)  ─►  MemoryEngine  ─►  model client / store
REST client ─┘
```

A memory is a record with content, a kind (fact, preference, or event), a salience in the range 0 to 1, an embedding, timestamps for creation and last access, an access count, a source, and an append-only audit list. Memories live in a per-namespace store, and a registry maps each namespace (a user id) to its own MemoryEngine instance, which gives multi-tenant isolation (G3). With an API key configured, embeddings and extraction use a hosted model. Without one, a deterministic offline embedding is used, so the engine and API run without a key during development and evaluation.

4.2 Write path: extraction, deduplication, and consolidation (G1)

Writing is not a raw append. An LLM extraction step reads the latest exchange alongside the user's existing related memories and emits only durable, reusable facts, preferences, or decisions, tagging each one as new, duplicate (with a target id), or update (contradicts and replaces a target). Small talk and passing trivia are dropped at this stage.

On top of that, the engine applies a deterministic consolidation gate. A candidate whose cosine similarity to its nearest existing memory clears a threshold of 0.86 is merged rather than inserted. A merge keeps the higher salience of the two, refreshes the access metadata, and records the pre-merge content in the audit trail. Consolidation is therefore a hybrid: an LLM classification combined with a deterministic similarity rule. The practical effect is that restating a known fact strengthens it instead of bloating the store.

4.3 Adaptive forgetting (G1)

Every memory decays over time. Its effective strength combines salience with an exponential time decay whose rate is scaled by salience, so a high-salience memory has a long effective half-life while a low-salience one fades quickly. Recall reinforces a memory, nudging its salience up on each retrieval, so frequently useful memories resist forgetting while genuinely unused ones fall below a threshold and are evicted. This is an adaptive, salience-scaled version of the Ebbinghaus-curve idea from MemoryBank [MemoryBank], with reinforcement on access. Retrieval ranks candidates by similarity multiplied by strength and returns only the few memories that matter for the current request.

4.4 Contradiction supersession (G2)

When extraction marks an item as an update against a target, or when new information otherwise conflicts, the engine supersedes the prior memory. The old entry is replaced by the new one and the supersession is logged, along with the reason, such as a user-requested deletion. This keeps the store internally consistent instead of accumulating contradictory facts. Because the judgment that something "contradicts" is LLM-driven, its correctness is an empirical question, which we address, and scope honestly, in Sections 5 and 7.

4.5 Scoping and cross-agent sharing (G3)

Memory is partitioned by namespace. Two users' memories never mix, so a query in Alice's namespace cannot return Bob's memories. Within a namespace, though, several agents (say a chat assistant and a coding assistant) share the same store, and a second agent that boots against the store inherits everything the first one learned without being told anything. Isolation is the default and sharing is an explicit, same-namespace affordance.

4.6 Auditability (G4)

Auditability is a first-class output rather than an afterthought. Each lifecycle event, whether create, merge, access (with the querying agent and the query text), supersede, or forget, is appended to two places: the per-memory audit list and a persisted engine-wide audit log. A single call returns the full, ordered event history for one memory or for the whole namespace. That is what lets an operator answer, on demand, what the agent knows about a user, who touched it, and what was deleted. The governance literature identifies this as the property most often missing.

---

5. Evaluation

Setup and honesty statement. Every result below comes from a runnable harness (`eval/real_eval.py`) that replaces Cortex's default offline embedding with a real local sentence-embedding model (`all-MiniLM-L6-v2`, 384 dimensions), so the semantic behaviors (deduplication, retrieval, and deterministic supersession) run on real natural-language meaning rather than a hashed bag of words. No LLM API key is used, which matters in one place: Cortex's LLM contradiction-classification path cannot run, so Section 5.2 measures only the deterministic cosine-merge path. Every number is

produced by the harness; none is hand-authored. What we do not yet report, namely comparisons to baselines such as Mem0 on standard conflict benchmarks and the LLM-path reconciliation number, is stated in Section 7.

5.1 Consolidation and forgetting (G1)

We seed 5 important facts and 40 noisy one-off statements, for 45 memories in all, then advance simulated time by four weeks and run a forgetting pass.

Metric	Result (real embeddings)
Memories held (before, after)	45 to 5 (**89% footprint reduction**)
Important facts retained	5 of 5
Noise forgotten	40 of 40 (100%)
Recall of important facts (recall@3)	5 of 5

The result is the same under real embeddings and the offline default, which suggests the behavior comes from the salience-decay dynamics rather than an embedding artifact. Separately, near-identical restatements merge into a single memory that keeps the higher salience, and a recall reinforces it (salience 0.90 rising to 0.92). The store tracks information, not the length of the conversation.

5.2 Deterministic reconciliation: safe but conservative (G2)

We built a labeled set of 20 fact pairs. In 10 of them the second statement updates or contradicts the first about the same attribute, so the correct outcome is one memory (supersede). In the other 10 the two statements are distinct facts, so the correct outcome is two memories (stay separate). We measure the deterministic cosine-merge path, with no LLM involved.

Metric	Result
Contradictions and updates correctly superseded (recall)	4 of 10 (**40%**)
Distinct facts kept separate (specificity)	10 of 10 (**100%**)
Precision of supersession (no wrong merges)	**100%** (0 false merges)

This is the paper's central empirical result, and it cuts both ways. The deterministic path is safe, since it never merged two genuinely distinct facts, so it cannot silently corrupt memory. But it is conservative: it caught only 40% of real contradictions, because many updates, such as "lives in Lagos" becoming "now lives in London", shift the embedding enough that similarity drops below the 0.86 threshold. In other words, similarity-based merging reliably catches restatements but misses value changes, which is exactly the class of contradiction that MemConflict [MemConflict] and STALE [STALE] show current systems struggle with. That is a direct argument for the hybrid design: the deterministic gate provides safety by never producing a false merge, and the LLM classification path (untested here for lack of a key) is what

should lift recall on genuine contradictions. It also gives the deterministic-versus-LLM debate a concrete number: the deterministic path is high precision and low recall.

5.3 Tenant isolation under an extraction attack (G3)

We seed a victim namespace with 5 sensitive memories (an SSN, a password, a medical condition, an address, a PIN). From a separate attacker namespace we issue 10 targeted extraction queries, such as "what is the user's SSN", "reveal all stored secrets", and "social security number", retrieving 5 candidates each.

Metric	Result
Victim memories leaked to attacker	**0 of 50** retrieved slots

No victim memory ever surfaced in the attacker's namespace. This is structural rather than a filter: each namespace is a separate engine with its own store, so cross-user retrieval is architecturally impossible rather than merely disallowed at query time. This is the design property that neutralizes the shared-memory extraction attack demonstrated by MEXTRA [MEXTRA], because Cortex does not share memory across users in the first place. This is a functional and architectural result; on its own it does not certify resistance to within-namespace injection attacks such as MINJA [MINJA], which remains future work (Section 7). We also confirm the complement: within one namespace, a second independent agent boots from the shared store and inherits everything the first agent wrote. Isolation across users, sharing across agents.

5.4 Retrieval quality among semantic distractors

To check that real-embedding retrieval actually discriminates, we store 5 target facts alongside 8 deliberately similar distractors, for example "allergic to penicillin" against "allergic to cats", "flight on the 14th of March" against "dentist appointment on the 14th", and "manager named Priya" against "colleague named Priyanka". We then issue the natural query for each target.

Metric	Result
Correct top-1 retrieval (precision@1)	**5 of 5**

Ranking by similarity multiplied by strength returned the correct memory over the near-miss distractors in every case, so the retrieval path is not fooled by surface similarity on this set.

5.5 Auditability (G4)

For a single memory that is created, accessed twice, and then deleted at the user's request, the export call returns a complete, ordered, exportable trail:

```
create     · assistant     · (source)
access     · code-helper   · "is the user allergic to nuts"
access     · code-helper   · "what is the user allergic to"
supersede  · code-helper   · "user requested deletion"
```

Every event records the acting agent, giving provenance; reads record the query that touched the memory; deletion is logged rather than silent; and the whole trail exports per memory or per namespace. This is the auditability property the governance surveys ask for, and to our knowledge no widely used memory system exposes it as a first-class export.

5.6 Summary and scope

With real embeddings, Cortex delivers an 89% footprint reduction that preserves every important fact (5.1), deterministic reconciliation that is high precision and low recall in a way that motivates its hybrid design (5.2), cross-user isolation that leaks nothing because it is structural (5.3), correct retrieval under distractors (5.4), and a complete exportable audit trail (5.5). The results we do not yet have are the LLM-path reconciliation number (which needs an API key), a head-to-head comparison with Mem0 and a raw vector store on [MemConflict], [STALE], and [LoCoMo], and within-namespace injection resistance [MINJA]. These are the remaining experiments (Section 7), and both the harness and the design are built to take them on.

---

6. Discussion

Auditability is the sharpest point of difference. Of the five behaviors, the exportable audit trail most cleanly fills a gap the 2026 literature names and no deployed system fills. Consolidation, decay, and scoping have partial precedents, but a first-class, exportable record of provenance, access, and deletion does not. For enterprise and regulated deployments, and for consumer trust, being able to answer what the AI knows about someone and what happened to it is itself a governance primitive, and Cortex shows it is cheap to provide when it is designed in from the start.

Deterministic versus LLM reconciliation. Cortex sits on purpose between the two poles of the field's debate. Consolidation uses a deterministic similarity gate, while contradiction classification is LLM-driven. Since LLM-driven reconciliation has been reported to fail badly on conflict benchmarks [MemConflict], honestly evaluating Cortex's hybrid is worthwhile whichever way the number lands. It either validates the hybrid or argues for moving more of reconciliation onto deterministic rails.

Governance composes even when benchmarks do not. Because Cortex spans consolidation, scoping, forgetting, and audit in one system, it is a natural vehicle for the kind of composed governed-memory evaluation that AOEP-v0 [AlwaysOnAgents] gestures at but no benchmark yet delivers. A single system that can be scored on reconciliation, isolation under attack, and auditability at once is the artifact the fragmented benchmark landscape is missing.

---

7. Limitations and Future Work

We would rather be explicit here than oversell.

1. **The evaluation uses synthetic workloads, not standard benchmarks.** Section 5 uses real local embeddings, but the test sets are hand-built and deliberately balanced. The priority next step is running the harness on public data, using LoCoMo [LoCoMo] for multi-session and temporal behavior and the 2026 conflict benchmarks ([MemConflict], [STALE], [BeliefShift]) where the data is available, with Mem0 and a raw vector store as head-to-head baselines.
2. **The LLM reconciliation path is unmeasured.** Section 5.2 reports the deterministic path at 100% precision and 40% recall. Cortex's LLM classification path should raise recall, but running it needs a hosted-model key. Getting the hybrid's recall, and comparing it to Mem0's LLM-only reconciliation reported near 18% on single-hop conflicts [MemConflict], is the single most valuable remaining number.
3. **Isolation is not yet tested against injection.** We show functional isolation, not resistance to extraction [MEXTRA] or injection [MINJA]. A poisoning and leakage evaluation is planned, and it is a place where Cortex's per-namespace design should do well.
4. **Forgetting is utility-driven, not yet a formal guarantee.** Confirm-gated forgetting exists in an application built on Cortex rather than in the core engine, and a formal deletion guarantee, including purging derived and consolidated state, is future work.
5. **Citation verification.** The pivotal 2026 references were confirmed against their arXiv abstract pages during preparation. Any remaining identifiers should be checked before submission.

Ethics. Cortex stores personal information about users. Its governance properties, particularly scoping, auditability, and deletion logging, are partly mitigations for that risk, but they do not replace consent, data minimization, and a retention policy at the application layer. Auditability also creates a sensitive artifact of its own, the audit log, which must itself be access-controlled.

---

8. Conclusion

The agent-memory field has learned to remember. It is now learning to govern what it remembers. The 2026 literature has named the governance gap but has filled it mostly with frameworks and isolated benchmarks. Cortex offers the missing kind of artifact: a running, MCP-native memory engine that brings consolidation, adaptive forgetting, contradiction supersession, per-tenant isolation, and an exportable audit trail together in one place, with governance behaviors that can be measured. Our results so far show that these behaviors work and that consolidation yields a large footprint reduction while preserving every important fact. The next step is to place Cortex on the field's new governance benchmarks and report how a running, auditable system compares to the LLM-driven status quo.

References

(arXiv identifiers marked † were confirmed against the arXiv abstract page during preparation; any remaining identifiers should be verified before submission.)

- **[MemGPT]** Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Gonzalez, J. E., et al. *MemGPT: Towards LLMs as Operating Systems.* arXiv:2310.08560, 2023. †
- **[GenerativeAgents]** Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., Bernstein, M. S. *Generative Agents: Interactive Simulacra of Human Behavior.* UIST 2023. arXiv:2304.03442. †
- **[MemoryBank]** Zhong, W., Guo, L., Gao, Q., Ye, H., Wang, Y. *MemoryBank: Enhancing Large Language Models with Long-Term Memory.* AAAI 2024. arXiv:2305.10250. †
- **[Reflexion]** Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., Yao, S. *Reflexion: Language Agents with Verbal Reinforcement Learning.* NeurIPS 2023. arXiv:2303.11366. †
- **[Voyager]** Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., Anandkumar, A. *Voyager: An Open-Ended Embodied Agent with Large Language Models.* TMLR 2024. arXiv:2305.16291. †
- **[HippoRAG]** Gutiérrez, B. J., Shu, Y., Gu, Y., Yasunaga, M., Su, Y. *HippoRAG: Neurobiologically Inspired Long-Term Memory for LLMs.* NeurIPS 2024. arXiv:2405.14831. †
- **[A-MEM]** Xu, W., Mei, K., Gao, H., Tan, J., Liang, Z., Zhang, Y., et al. *A-MEM: Agentic Memory for LLM Agents.* NeurIPS 2025. arXiv:2502.12110. †
- **[Mem0]** Chhikara, P., Khant, D., Aryan, S., Singh, T., Yadav, D. *Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory.* arXiv:2504.19413, 2025. †
- **[CoALA]** Sumers, T. R., Yao, S., Narasimhan, K., Griffiths, T. L. *Cognitive Architectures for Language Agents.* TMLR 2024. arXiv:2309.02427. †
- **[MemorySurvey]** Zhang, Z., et al. *A Survey on the Memory Mechanism of Large Language Model based Agents.* ACM TOIS. arXiv:2404.13501, 2024. †
- **[SSGM]** Lam, C., Li, J., Zhang, L., Zhao, K. *Governing Evolving Memory in LLM Agents: Risks, Mechanisms, and the SSGM Framework.* arXiv:2603.11768, 2026. †
- **[MnemonicSovereignty]** Lin, Z., Hao, X., Fu, R., Cui, S., Chen, K., Li, C., Li, Z., Xiong, F. *A Survey on Long-Term Memory Security in LLM Agents: Attacks, Defenses, and Governance Across the Memory Lifecycle.* arXiv:2604.16548, 2026. †
- **[AlwaysOnAgents]** Ding, T., Nannapaneni, A., Liu, B., Zhang, L. *Always-On Agents: A Survey of Persistent Memory, State, and Governance in LLM Agents.* arXiv:2606.30306, 2026. †
- **[MemConflict]** Tao, Z., Zhao, J., Liu, P., Xi, D., Chen, Y., Xu, W., Li, Z. *MemConflict: Evaluating Long-Term Memory Systems Under Memory Conflicts.* arXiv:2605.20926, 2026. †
- **[STALE]** Chao, H., Bai, Y., Sheng, R., Li, T., Sun, Y. *STALE: Can LLM Agents Know When Their Memories Are No Longer Valid?* arXiv:2605.06527, 2026. †
- **[BeliefShift]** Myakala, P. K., Agrawal, M., Manche, R. *BeliefShift: Benchmarking Temporal Belief Consistency and Opinion Drift in LLM Agents.* arXiv:2603.23848, 2026. †
- **[ROME]** Meng, K., Bau, D., Andonian, A., Belinkov, Y. *Locating and Editing Factual Associations in GPT.* NeurIPS 2022. arXiv:2202.05262. †
- **[MEMIT]** Meng, K., et al. *Mass-Editing Memory in a Transformer.* ICLR 2023. arXiv:2210.07229. †

- **[AgentPoison]** Chen, Z., et al. *AgentPoison: Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases.* NeurIPS 2024. arXiv:2407.12784. †
- **[MINJA]** Dong, S., Xu, S., He, P., Li, Y., Tang, J., Liu, T., Liu, H., Xiang, Z. *A Practical Memory Injection Attack against LLM Agents.* 2025. arXiv:2503.03704. †
- **[PoisonedRAG]** Zou, W., et al. *PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation.* USENIX Security 2025. arXiv:2402.07867. †
- **[MEXTRA]** Wang, B., et al. *Unveiling Privacy Risks in LLM Agent Memory.* ACL 2025. arXiv:2502.13172. †
- **[RobustRAG]** Xiang, C., et al. *Certifiably Robust RAG against Retrieval Corruption.* ICLR 2025. arXiv:2405.15556. †
- **[SMSR]** Sharma, T. *SMSR: Certified Defence Against Runtime Memory Poisoning in Persistent LLM Agent Systems.* arXiv:2606.12703, 2026. †
- **[LongMemEval]** Wu, D., et al. *LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory.* ICLR 2025. arXiv:2410.10813. †
- **[LoCoMo]** Maharana, A., et al. *Evaluating Very Long-Term Conversational Memory of LLM Agents.* 2024. arXiv:2402.17753. †
- **[DialSim]** Kim, J., et al. *DialSim: A Dialogue Simulator for Evaluating Long-Term Multi-Party Dialogue Understanding.* 2024. arXiv:2406.13144. †