FROM DELETION TO ACTION: AN EMPIRICAL STUDY OF SELECTIVE DELETION IN AGENT MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Persistent memory enables large language model (LLM) agents to draw on past interactions. Privacy and other practical needs motivate selective deletion: removing designated information and its influence on subsequent behavior while preserving other useful memories. Yet it remains unclear how well the record-level deletion operations available in existing memory frameworks satisfy this requirement. We empirically assess selective deletion of user preferences across four memory frameworks by tracing deletion operations through stored memory and context to task actions. We measure behavioral residue relative to a NeverSeen baseline with no prior exposure to the target preference. We find that a mismatch between preferences and records can leave alternative representations of the target preference intact or remove non-target preferences consolidated in the same record. Moreover, failure to detect a target preference in context neither establishes its removal from stored memory nor guarantees its absence from later contexts. At the behavioral level, near-zero overall mean residue can mask positive residue in subgroups where the target preference remains detectable. We find that post-deletion behavioral residue varies across reading model configurations within the same memory framework and on the same scenarios. These findings show that record deletion, absence from context, and the disappearance of behavioral influence need not coincide, motivating a joint assessment of stored memory, context visibility, and subsequent task actions.

1 INTRODUCTION

Persistent memory enables agents to retain user information across interactions and personalize subsequent tasks (Zhong et al., 2024; Salemi et al., 2024). A preference expressed in one session can therefore influence choices in later sessions. Users may ask a system to stop storing particular information for privacy reasons or to stop acting on a preference that has changed. These requests have a specific scope: remove the designated information and its influence on subsequent behavior while preserving other useful memories. We refer to this requirement as *selective deletion*.

Satisfying this requirement is complicated by a mismatch between user preferences and the objects that memory frameworks delete. A user specifies a preference, but a framework operates on records, summaries, or other memory representations. One preference may be represented in several records, while a single record may consolidate multiple preferences. Deleting an identified record may therefore leave other representations of the target preference, whereas deleting a consolidated record may also remove non-target preferences. This raises an empirical question: how well do the deletion operations available in existing memory frameworks satisfy a user's request for selective deletion?

Recent studies examine whether deleting agent memory removes its downstream influence. Chen et al. (2026) evaluate deletion through utility and information extraction, while MemLeak examines the recoverability of deleted information in text and image memory (Wang & Zhang, 2026). MemScope measures behavioral residue in tool actions and compares source-record deletion with the removal of derived content (Liu & Li, 2026). Building on these studies, we examine what different deletion scopes remove and preserve in *stored memory*, whether retained target preferences appear in *context*, and how their presence in context relates to *task actions*. We consider both the removal of target preferences and the retention of non-target preferences.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

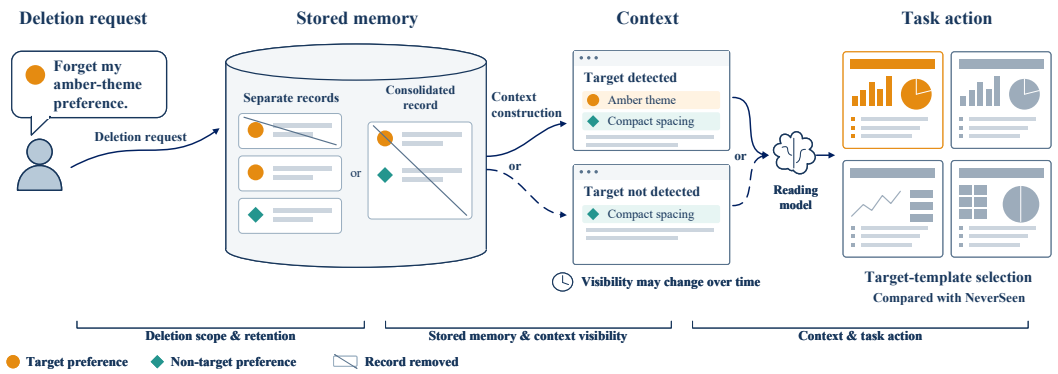

Figure 1: From deletion to action. We examine the relationships between deletion request and stored memory, stored memory and context, and context and task action.

Specifically, we study Memobase (memodb-io, n.d.), Mem0 (Chhikara et al., 2025), LangMem (langchain-ai, n.d.), and Zep (Rasmussen et al., 2025) using dashboard deployment tasks spanning 12 preference dimensions. Each scenario includes a target preference to delete and a designated non-target preference to preserve. After each framework forms memory from the interaction history, we apply deletion operations with specified scopes, inspect stored memory and context, and record the agent's subsequent template selection and deployment. We measure *behavioral residue* as the difference in target-template selection rates between a post-deletion condition and a *NeverSeen* baseline with no prior exposure to the target preference. This comparison accounts for target-template selection without prior exposure to the preference. We additionally compare language model configurations and track changes in target visibility over time after deletion in Zep. Figure 1 shows the chain and the observations collected at each stage.

Our experiments yield three findings along this chain:

- **Deletion scope affects both target removal and non-target retention.** In Memobase, broad-scope deletion reduces the number of scenarios with the target preference detected in context from 137 to seven; the designated non-target preference is also no longer detected in eight scenarios where it was previously present. In LangMem, targeted deletion removes the sole memory record in all 125 eligible scenarios; in 115 of these scenarios, that record also contains the designated non-target preference.
- **A stored preference may go undetected in context, and its visibility in context may change over time.** After targeted deletion in Zep, the target preference remains in stored memory in 66 of 75 scenarios but is not detected in any of the 75 contexts. Separate temporal experiments show that a target preference detected shortly after targeted deletion is no longer detected at later checkpoints, whereas after source-thread deletion, a target preference initially undetected in context reappears.
- **Behavioral residue varies across context states and language model configurations.** After broad-scope deletion in Memobase, mean behavioral residue is 17.6 percentage points among the seven scenarios where the target preference is still detected in context, despite an overall mean of 1.0 percentage point. Across three language model configurations evaluated on the same 30 Memobase scenarios, residue after targeted deletion ranges from 18.1 to 72.0 percentage points.

2 Related Work

Deletion in agent memory. Recent studies evaluate deletion in agent memory through information retention and subsequent behavior. Chen et al. (2026) combine personalized utility, extraction risk, and residual-information checks across memory layers. They find that derived summaries can retain target information after the original records are deleted and compare re-summarization with complete removal. MemLeak uses information provenance to examine how retained text and images support the recovery of deleted facts, with behavioral and temporal probes in pilot experiments

(Wang & Zhang, 2026). MemScope measures behavioral residue in tool actions against a baseline without exposure to the target source and compares source-record deletion with the removal of derived content (Liu & Li, 2026). These studies establish the need to assess information recoverability and behavioral influence after record deletion. Building on this work, we jointly examine deletion scope, stored memory, context, and task actions, considering both target removal and non-target retention.

Memory organization, access, and evaluation. Agent memory systems transform interaction histories into different memory representations. MemoryBank and MemoryOS support long-term interaction through memory management and hierarchical organization, respectively (Zhong et al., 2024; Kang et al., 2025). Generative Agents consolidate experiences into higher-level reflections, while A-Mem organizes memory through links between notes and updates to existing representations (Park et al., 2023; Xu et al., 2025). These representations determine the objects on which deletion operates, motivating examination of how the information a user asks to delete maps onto stored memory objects.

Stored information must also be accessed and used in the current task. LongMemEval analyzes memory systems through indexing, retrieval, and reading (Wu et al., 2025). Lost in the Middle examines how the position of relevant information in context affects its use (Liu et al., 2024). What Eviction Destroys combines checks of retained evidence with restoration counterfactuals to distinguish storage loss, access failure, and failures to use restored evidence under capacity-driven eviction (Shen, 2026). These studies motivate distinguishing stored memory, context, and task performance. Our post-deletion assessment distinguishes retained content, returned context, and subsequent task actions.

Memory is evaluated through a range of tasks. LaMP evaluates personalized classification and generation, while LoCoMo assesses question answering, event summarization, and multimodal generation over long conversations (Salemi et al., 2024; Maharana et al., 2024). MemoryAgentBench evaluates selective forgetting through single- and multi-hop questions that require using updated facts (Hu et al., 2026). Memora includes recommendation and penalizes the use of outdated or invalid information (Uddin et al., 2026). ImplicitMemBench measures behavioral adaptation through procedural memory, priming, and conditioning (Qin et al., 2026), while Xiong et al. (2026) examine how adding and removing memories relates to experience-following behavior and error propagation. We evaluate whether an agent continues to act on the preference designated for deletion through its subsequent task choices.

Target removal and non-target retention. Machine unlearning studies the removal of training-data influence from learned models (Cao & Yang, 2015). TOFU constructs forget and retain tasks using fictitious author profiles, while MUSE evaluates forgetting, privacy leakage, utility preservation, scalability, and sequential unlearning requests (Maini et al., 2024; Shi et al., 2025). These evaluations emphasize assessing target removal alongside the preservation of useful information and capabilities.

We apply this evaluation principle to user preferences in external memory, checking both target removal and the retention of a designated non-target preference. Our deletion operations act on external memory objects while leaving model parameters unchanged. We examine their effects on stored memory, context, and task actions.

3 Experimental Setup

3.1 Dataset and Task

We study selective deletion using a dataset of 138 constructed dashboard deployment scenarios spanning 12 preference dimensions: theme color, layout spacing, information density, refresh interval, font size, tooltip behavior, sidebar position, widget style, chart type, date format, panel border, and number format. Each scenario pairs an interaction history with subsequent deployment tasks. The history contains a target preference to delete, a designated non-target preference to preserve, and unrelated filler interactions.

Table 1: Memory structures, deletion operations, and context sources in the evaluated configurations.

Framework	Memory structure	Targeted deletion	Broad-scope deletion	Context source
Memobase	Profiles and events	Delete the target profile entry	Delete the target profile entry and target-related events	User context
Mem0	Memory entries	Delete one identified target entry	Clear the user's memory	User memory entries
LangMem	Entries in a namespace	Delete the entry containing the target preference	—	Manager search results
Zep	Episodes, graph edges, and user-level representations	Apply object deletion interfaces to target-matched episodes, edges, and nodes	Delete the source thread	User context block

The interaction history is first written to each framework to form memory. After a deletion operation, the agent receives context from the framework and a new project request. It then selects and deploys a dashboard template in a simulated tool environment by listing candidates, inspecting their configurations, and submitting a deployment. The project request asks the agent to complete the deployment rather than explicitly report the target preference.

For example, the history may express a preference for an amber theme, while the subsequent request asks the agent to deploy a dashboard for a new project. Each behavioral probe provides four candidate templates, exactly one of which has the target attribute value. Template configurations are available through tools. We record whether the agent ultimately selects and deploys the amber-themed template.

3.2 Memory Structures and Deletion Operations

We evaluate Memobase, Mem0, LangMem, and Zep against two reference conditions. Seen forms memory from a history containing the target preference, while NeverSeen uses the corresponding history without exposure to that preference. Starting from memory formed with the target preference, we apply targeted deletion by locating target-related records and invoking the framework's deletion interfaces.

We additionally examine broad-scope deletion in Memobase, Mem0, and Zep. These operations act at different scopes: Memobase removes target-related events in addition to the target profile entry, Mem0 clears the user's memory, and Zep deletes the source thread. In our experiments, targeted deletion in LangMem already removes the sole memory entry, so clearing the store would yield the same post-deletion state. Table 1 summarizes the memory structures, deletion operations, and context sources.

The behavioral conditions are paired by scenario. Memobase applies targeted deletion and then broad-scope deletion sequentially to the same user. LangMem compares the same store before and after targeted deletion. Mem0 and Zep form memory separately for Seen, NeverSeen, and each deletion condition evaluated for deployment choices. During Zep's targeted deletion, attempts to delete the canonical user node are refused by the interface, so this node remains.

For Zep, we evaluate deployment choices only after targeted deletion, using context returned for the source thread in which the history was written. Separate temporal experiments compare targeted deletion with source-thread deletion by reading context from a second thread associated with the same user. These experiments form memory independently for each deletion operation.

3.3 Models and Behavioral Evaluation

For the deployment-choice experiment across the four frameworks, we use Qwen2.5-7B-Instruct as the reading model. Memobase, Mem0, and LangMem also use this model to form memory, while Zep uses its hosted extraction process. We obtain context from the sources listed in Table 1

and provide it to the reading model together with the project request. The model performs the deployment task through the common tool environment.

Each scenario contains six behavioral probes that vary the candidate templates, their order, and accompanying metrics. We sample five trajectories per probe at a temperature of 0.7, yielding 30 trajectories per scenario and condition. All deployment probes and repeated samples for a given scenario and condition use the same context. Conditions share the same probe set and sampling-seed schedule.

We also compare Qwen2.5-7B-Instruct, Mistral-Small-24B-Instruct-2501 (Mistral-24B), and Qwen2.5-32B-Instruct on the same 30 Memobase scenarios. We reuse memory states formed with Qwen2.5-7B-Instruct and vary only the language model configuration used to read memory and perform the task under the corresponding deletion conditions.

Implementation configuration. Memobase is self-hosted with Python client 0.0.27. Mem0 uses `mem0ai` 2.0.20 with FAISS and `BAAI/bge-small-en-v1.5` embeddings. LangMem 0.0.30 uses `InMemoryStore` with `enable_deletes=True`. Zep uses the hosted Cloud API. Qwen2.5-7B-Instruct and Qwen2.5-32B-Instruct run in BF16; Mistral-24B uses `RedHatAI/Mistral-Small-24B-Instruct-2501-FP8-dynamic`, with FP8 weights and dynamic FP8 activations. The 7B reader is served with vLLM 0.28.0.

3.4 Information States and Evaluation Measures

Information states. We assess the target preference separately in stored memory and context. Storage detection and context detection refer to matches in inspectable records and in the text supplied to the reading model, respectively. Checks use case-insensitive matching of preference values, with word-boundary or dimension-specific constraints, and cover references within descriptions of other attributes. We measure non-target retention at the same observation level before and after deletion: among scenarios where the designated non-target preference was detected before deletion, we compute the proportion where it is still detected afterward. When paired pre-deletion observations are unavailable, we report the post-deletion detection count.

Behavioral scoring and comparison sets. A valid deployment of the target template is scored as 1, and a valid deployment of another template as 0. Trajectories without a valid deployment are unscorable. For each scenario and condition, the target-template selection rate is the proportion of scorable deployments selecting the target template. For each framework and condition, the valid scenario set consists of scenarios with at least one scorable deployment. A deletion condition additionally requires detection of the target preference before deletion. Within each framework, the comparison set for two conditions is the intersection of their valid scenario sets. The comparison sets used for deployment-choice evaluation contain 137, 137, 125, and 75 scenarios for Memobase, Mem0, LangMem, and Zep, respectively. Further scoring details are provided in Appendix A.

Behavioral residue. Let $p_{i,a}$ denote the target-template selection rate in scenario i under condition a. The mean paired difference between conditions a and b is

$$\Delta_{a,b} = \frac{1}{|\mathcal{I}_{a,b}|} \sum_{i \in \mathcal{I}_{a,b}} (p_{i,a} - p_{i,b}), \tag{1}$$

where $\mathcal{I}_{a,b}$ is the intersection of the valid scenario sets for conditions a and b within the framework being evaluated. Seen minus NeverSeen measures the exposure effect, while a post-deletion condition minus NeverSeen measures behavioral residue. We report $100\Delta_{a,b}$ in percentage points. Because the model can select the target template without prior exposure to the target preference, residue measures the post-deletion difference relative to this baseline. The retained-effect ratio divides behavioral residue by the exposure effect, with both differences computed for the same model configuration and scenario set. For targeted deletion, we report this ratio as a percentage:

$$R = \frac{\Delta_{\text{Targeted, NeverSeen}}}{\Delta_{\text{Seen, NeverSeen}}} \times 100\%. \tag{2}$$

Table 2: Target and designated non-target preference states after deletion. Non-target presence reports post-deletion detection among all evaluated scenarios. Retention restricts the denominator to scenarios with pre-deletion detection at the same observation level. Memobase broad-scope retention is measured relative to the preceding targeted-deletion state. Mem0 target detection in context is inferred from retained entries.

Framework / operation	Stored memory after deletion	Target detected in context	Non-target preference
Memobase / targeted	Target profile entry removed; target preference remains in events	137/137	Presence: 137/137
Memobase / broad-scope	Target profile entry and target-related events removed; some cross-attribute references remain	7/137	Retention: 129/137
Mem0 / targeted	Identified target entry removed; other expressions of the target preference remain in eight scenarios	8/137	Presence: 136/137
Mem0 / broad-scope	All user memory entries removed	0/137	Presence: 0/137
LangMem / targeted	Sole memory entry removed; store empty in all 125 scenarios	0/125	Retention: 0/115
Zep / targeted	Target preference remains in the canonical user node in 66/75 scenarios	0/75	Retention: 73/73

Statistical estimation. Scenarios receive equal weight. We compute 95% confidence intervals using 10,000 paired bootstrap resamples at the scenario level, preserving each scenario's observations across the compared conditions. Residual subgroups are defined post hoc from the information checks. Within each subgroup, we compute the mean paired difference between the deletion and NeverSeen conditions, holding subgroup membership fixed during resampling.

4 Results and Analysis

We examine selective deletion at three levels: stored memory, context, and task actions (Figure 1). We first assess whether deletion removes the target preference while preserving the designated non-target preference. We then examine whether preferences retained in storage appear in context and how context visibility changes over time. Finally, we measure behavioral residue in deployment choices and compare it across context states and language model configurations.

4.1 Deletion Scope and Preference Retention

We first examine how deletion scope relates to target and non-target retention in stored memory. Table 2 summarizes the post-deletion state of stored memory, target detection in context, and the status of the designated non-target preference.

Deleting an identified record may leave other expressions of the target preference. In Memobase, deleting the target profile entry leaves events that carry the target preference into context in all 137 scenarios. Broad-scope deletion additionally removes target-related events, reducing target detection in context to seven scenarios. In these remaining cases, descriptions of other attributes still refer to the target preference. In Mem0, deleting one identified target entry leaves other expressions of the same preference in eight scenarios.

Deletion can also remove preferences intended for preservation. In Memobase, broad-scope deletion makes the designated non-target preference undetectable in eight of the 137 scenarios where it was present after targeted deletion, leaving 129/137 retained. Mem0's broad-scope operation clears all user memory entries, so it removes the non-target preference along with the target: post-deletion non-target presence is 136/137 under targeted deletion and 0/137 under broad-scope deletion. Non-target loss also occurs under targeted deletion when a single record contains both preferences. In LangMem, each of the 125 eligible scenarios has a sole memory entry before deletion, and 115 of these entries contain both preferences. Deleting that entry removes both preferences in those 115 scenarios, yielding non-target retention of 0/115.

Table 3: Target detection in new-thread context for seven matched Zep scenarios. Entries report detected scenarios / eligible scenarios at scheduled checkpoints, timed from procedure start for targeted deletion and from API-call completion for source-thread deletion.

Deletion operation	30-second checkpoint	300-second checkpoint
Targeted deletion	2/7	0/7
Source-thread deletion	0/7	7/7

In these configurations, a preference may span several records, leaving expressions outside the deletion scope, while a single record may contain multiple preferences that are removed together.

4.2 From Stored Memory to Context Visibility

We next examine whether target preferences retained in stored memory appear in the context returned to the agent, and how their visibility changes over time. In the Zep deployment-choice experiment, the target preference remains in the canonical user node after targeted deletion in 66 of 75 scenarios, but it is not detected in any of the 75 contexts returned for the source thread (Table 2). In LangMem, the target preference is also not detected in context, but the inspected stores are empty. Thus, checking context alone cannot distinguish between a preference that has been removed from storage and one that remains stored but is not returned.

To examine changes in visibility over time, we conduct separate Zep experiments that read context from a second thread associated with the same user. We compare targeted deletion and source-thread deletion on seven matched scenarios where the target preference is detected before deletion under both operations. Memory is formed separately for each operation. Table 3 reports detection at the scheduled 30-second and 300-second checkpoints.

After targeted deletion, the target preference is detected in two scenarios at the early checkpoint, both in user summaries, and in none at the later checkpoint. It is also not detected at the additional 900-second checkpoint. After source-thread deletion, the pattern is reversed: the target preference is not detected at the early checkpoint but reappears in all seven scenarios at the later checkpoint. Additional source-thread experiments also show later reappearance (Appendix B).

Failure to detect a preference in context therefore establishes neither its removal from stored memory nor its absence from later contexts.

4.3 From Context to Task Actions

We then examine how post-deletion context relates to task actions, measuring behavioral residue in deployment choices relative to NeverSeen.

Context visibility and overall behavioral residue. Table 4 pairs target detection in context with selection rates and behavioral residue, using Seen and NeverSeen as reference conditions.

In Memobase, the target preference is detected in all 137 contexts after targeted deletion. Mean target-template selection is 44.2% after targeted deletion versus 26.4% under NeverSeen, with mean behavioral residue of 17.9 percentage points (95% CI: [14.3, 21.5]). After broad-scope deletion, target detection falls to seven scenarios, the selection rate falls to 27.4%, and mean residue falls to 1.0 percentage point (95% CI: $[-1.2, 3.2]$).

In Mem0, targeted deletion leaves target detection in eight scenarios and a mean residue of -0.5 percentage points; broad-scope deletion yields no target detection and a mean residue of -0.8 percentage points. LangMem and Zep likewise show no target detection in context after targeted deletion, with mean residues of 1.0 and -0.5 percentage points, respectively. Across these conditions, widespread target detection after Memobase targeted deletion accompanies substantial mean behavioral residue, whereas the conditions with limited or no target detection have near-zero overall means.

Table 4: Context visibility and task behavior after deletion. Target-template selection rates (%) are scenario means; residue is the mean paired difference from NeverSeen (percentage points), calculated before rounding. Conditions use the same comparison set within each framework, with equal weight per scenario. Context detection reports detected / evaluated scenarios. Mem0 context status is inferred from retained entries; Zep uses source-thread context.

Framework	Seen	NeverSeen	Deletion scope	Target in context	Post-deletion selection	Residue
Memobase	55.7	26.4	Targeted	137/137	44.2	+17.9
			Broad-scope	7/137	27.4	+1.0
Mem0	35.2	26.8	Targeted	8/137	26.3	−0.5
			Broad-scope	0/137	26.0	−0.8
LangMem	51.9	25.3	Targeted	0/125	26.3	+1.0
Zep	69.8	25.3	Targeted	0/75	24.8	−0.5

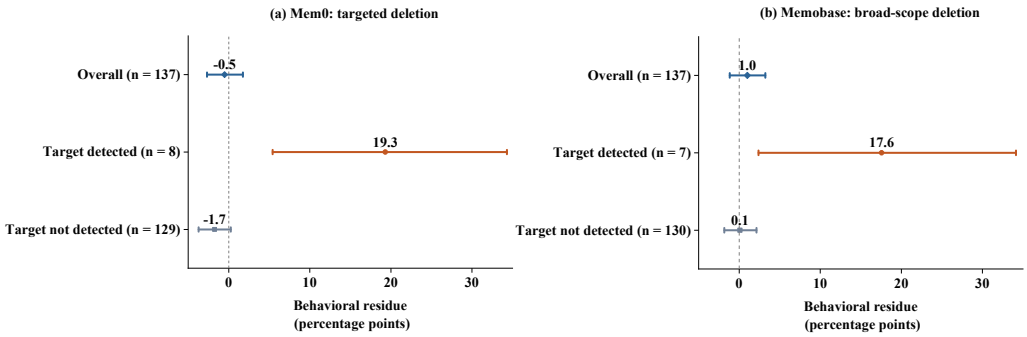

Figure 2: Behavioral residue overall and by target detection in context. Points show mean paired differences from each group's matched NeverSeen baseline; bars show 95% paired bootstrap confidence intervals. Mem0 context labels are inferred from retained entries.

Behavioral residue by context state. We further separate the Mem0 targeted-deletion and Memobase broad-scope-deletion results by whether the target preference is detected in context. Figure 2 reports behavioral residue for each subgroup relative to its scenario-matched NeverSeen baseline.

In the eight Mem0 scenarios with target detection, mean target-template selection exceeds the NeverSeen rate: 36.0% versus 16.7%. Mean residue is 19.3 percentage points (95% CI: [5.4, 34.3]), versus −1.7 percentage points (95% CI: [−3.7, 0.3]) in the remaining 129 scenarios. In the seven Memobase scenarios with target detection, mean selection exceeds the NeverSeen rate: 34.8% versus 17.2%. Mean residue is 17.6 percentage points (95% CI: [2.4, 34.1]), versus 0.1 percentage point (95% CI: [−1.8, 2.1]) in the remaining 130 scenarios. In both deletion conditions, the near-zero overall mean masks positive mean residue in the subgroup with target detection.

Behavioral residue across model configurations. Table 5 compares three reading model configurations on the same 30 Memobase scenarios. It reports residue after targeted and broad-scope deletion and, for targeted deletion, the proportion of each configuration's Seen–NeverSeen exposure effect retained.

After targeted deletion, mean residue ranges from 18.1 to 72.0 percentage points across the three configurations, corresponding to 51.4%–98.0% of their respective exposure effects. Qwen2.5-32B-Instruct retains nearly the full exposure effect, whereas Qwen2.5-7B-Instruct retains about half. After broad-scope deletion, mean residue is close to zero for all three configurations.

Table 5: Behavioral residue across three language model configurations on the same 30 Memobase scenarios. Residue is reported in percentage points with 95% confidence intervals. The retained-effect ratio is for targeted deletion.

Model	Targeted residue	Retained-effect ratio	Broad-scope residue
Qwen2.5-7B-Instruct	18.1 [10.3, 26.4]	51.4%	−3.3 [−7.3, 0.9]
Mistral-24B	39.1 [32.9, 44.0]	86.0%	−2.8 [−6.9, 0.8]
Qwen2.5-32B-Instruct	72.0 [67.2, 76.4]	98.0%	−1.3 [−4.8, 2.3]

5 Conclusion

We examine selective deletion across stored memory, context, and task actions in four memory frameworks. Deleting a target record can leave other expressions of the preference, while deleting a consolidated record can also remove non-target preferences. Retained target information may go undetected in context at one checkpoint and be detected at a later one. Behavioral residue varies across context states and model configurations, and near-zero overall means can mask positive residue in subgroups with target detection.

By relating deletion scope, target removal, and non-target retention to context visibility and subsequent task actions, our joint assessment distinguishes outcomes that look similar at a single stage.

References

Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pp. 463–480, 2015. doi: 10.1109/SP.2015.35.

Rachel Chen, Guilin Zhang, Kai Zhao, Xu Chu, Amine Anoun, and Jerry Ting. Deployment-time memorization in foundation-model agents. In *The Second Workshop on the Impact of Memorization on Trustworthy Foundation Models at ICML*, 2026. URL https://openreview.net/forum?id=mc7fVimV9b.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI 2025*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in LLM agents via incremental multi-turn interactions. In *International Conference on Learning Representations*, volume 2026, pp. 156259–156291, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/fd1eff9dd295df50a41f2521942fa31d-Paper-Conference.pdf.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

langchain-ai. LangMem. GitHub repository, n.d. URL https://github.com/langchain-ai/langmem. Accessed September 26, 2026.

Jiabei Liu and Mingyang Li. Deletion is not forgetting: Behavioral residuals of deleted memories in stateful LLM agents. SSRN preprint 7428966, 2026. URL https://papers.ssrn.com/sol3/papers.cfm?abstract_id=7428966.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 02 2024. ISSN 2307-387X. doi: 10.1162/tacl_a_00638. URL https://doi.org/10.1162/tacl_a_00638.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Pratyush Maini, Zhili Feng, Avi Schwarzschild, Zachary Chase Lipton, and J Zico Kolter. TOFU: A task of fictitious unlearning for LLMs. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=B41hNBoWLo.

memodb-io. Memobase. GitHub repository, n.d. URL https://github.com/memodb-io/memobase. Accessed September 26, 2026.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701320. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Chonghan Qin, Xiachong Feng, Weitao Ma, Xiaocheng Feng, and Lingpeng Kong. ImplicitMemBench: Measuring unconscious behavioral adaptation in large language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28232–28261, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1301. URL https://aclanthology.org/2026.acl-long.1301/.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. URL https://arxiv.org/abs/2501.13956.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. LaMP: When large language models meet personalization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7370–7392, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.399. URL https://aclanthology.org/2024.acl-long.399/.

Chen Shen. What eviction destroys: A restore-counterfactual audit of forgetting in agent memory. In *COLM Workshop on Context Beyond the Window: Persistent Knowledge in Language Models*, 2026. URL https://openreview.net/forum?id=8rOh73WoJh.

Weijia Shi, Jaechan Lee, Yangsibo Huang, Sadhika Malladi, Jieyu Zhao, Ari Holtzman, Daogao Liu, Luke Zettlemoyer, Noah A. Smith, and Chiyuan Zhang. MUSE: Machine unlearning six-way evaluation for language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=TArmA033BU.

Md Nayem Uddin, Kumar Shubham, Eduardo Blanco, Chitta Baral, and Gengyu Wang. From recall to forgetting: Benchmarking long-term memory for personalized agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26814–26841, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1337. URL https://aclanthology.org/2026.findings-acl.1337/.

Kuan Wang and Chao Zhang. MemLeak: Diagnosing information leaks in multimodal agent memory, 2026. URL https://arxiv.org/abs/2606.29788.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval:benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts LLM agents: An empirical study of experience-following behavior. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 623–645, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.27. URL https://aclanthology.org/2026.acl-long.27/.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: enhancing large language models with long-term memory. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

A Experimental and Evaluation Details

A.1 Sample Composition

Each framework starts with 138 candidate scenarios. Comparison sets contain scenarios with scorable deployments in both conditions; deletion conditions also require target detection before deletion. Table 6 distinguishes the scenarios with scorable Seen and NeverSeen outputs from those included in the main deletion comparisons.

Table 6: Scenario counts before and after eligibility checks for deletion comparisons.

Framework	Candidates	Seen	NeverSeen	Deletion comparison set
Memobase	138	137	137	137
Mem0	138	138	138	137
LangMem	138	138	138	125
Zep	138	137	137	75

For Memobase broad-scope deletion, behavioral outputs for the seven scenarios with target detection were collected in a separate batch using the same procedure, sampling temperature, and probe schedule as the other scenarios.

A.2 Task Execution and Scoring

The deployment task provides three tools: `list_templates` returns candidate template identifiers, `inspect_template` returns template configurations and information on accessibility, compatibility, latency, and resource cost, and `deploy_template` performs deployment. Model input combines context, project background, and the task instruction. The evaluator identifies which candidate matches the target preference; this designation is not included in model input.

Each trajectory allows up to eight model calls, with a maximum of 512 output tokens per call. The system prompt asks the model to inspect candidate templates and complete one deployment. Scoring uses the first valid deployment: selecting the target template is scored as 1, and selecting another template as 0. A trajectory containing invalid tool calls remains scorable if it subsequently completes a valid deployment. Trajectories without a valid deployment are excluded from the selection rate.

Across frameworks and conditions, unscorable trajectories account for approximately 0.12%–2.36% of observations. Every scenario with behavioral observations has at least one scorable deployment.

A.3 Information Checks

Information checks use case-insensitive lexical matching. Whole-word matching detects preference values as complete words rather than substrings of other words. Dimension-anchored matching additionally requires the value and a dimension-specific phrase to appear in the same text segment. For example, color preferences use "color theme," and chart preferences use "chart style" or "chart type." Segments are separated by periods, semicolons, or line breaks. Table 7 summarizes the observation sources and matching rules.

Table 7: Observation sources and information checks.

Framework	Observation source	Information checks
Memobase	Profile records and returned context	Whole-word matching for target and designated non-target preference values.
Mem0	Retained user entries, concatenated as model input	Dimension-anchored matching in saved entries; context status inferred from these entries.
LangMem	Namespace entries and manager search results	Dimension-anchored matching identifies the target entry and checks whether it contains the designated non-target preference; the store is inspected after deletion.
Zep	Retained graph objects and returned context	Target-matched objects are inspected after deletion; whole-word matching detects target and designated non-target preferences in context.

Deletion selection follows each operation's scope. Memobase locates the target profile by preference dimension and selects target-related events by whole-word matching. Mem0 and LangMem use dimension-anchored matching to select a target entry. Zep selects episodes, edges, and nodes using case-insensitive substring matching over their text fields.

B Zep Temporal Experiments: Setup and Extended Results

B.1 Experimental Setup and Sample Composition

For each scenario, we associate two threads with the same Zep user record. Thread A receives the history containing the target preference, while thread B receives a neutral request without it. For each deletion operation, we form memory independently and inspect the context returned for thread B. Source-thread deletion removes A while retaining the user record and B, allowing observation to continue after deletion.

Both experiments use the same 30 candidate scenarios. In the source-thread experiment, 28 meet the pre-deletion eligibility requirements. In the targeted-deletion experiment, we check memory formation every 30 seconds for up to 600 seconds, requiring the target to appear in both the user graph and the source-thread context, with the context no longer changing. Nine scenarios reach this state; seven also have the target detected in thread B's context before deletion. Table 8 summarizes scenario counts at each eligibility stage for the two deletion operations.

Table 8: Scenario eligibility for the two temporal experiments.

Stage	Source-thread deletion	Targeted deletion
Candidate scenarios	30	30
Target-formation requirement met	28	9
Target detected in thread B before deletion	28	7
Eligible under both operations	7	7

The main text compares the seven scenarios eligible under both operations. All eligibility checks use pre-deletion states.

Both operations have scheduled 30-second and 300-second checkpoints; targeted deletion additionally has a 900-second checkpoint. Timing starts at the beginning of the deletion procedure for targeted deletion and when the deletion call completes for source-thread deletion.

B.2 Extended Source-Thread Observations

Across all 28 eligible source-thread scenarios, the target is not detected in context at the early checkpoint but is detected in all scenarios at approximately five minutes.

For a further experiment, we select the first 12 scenarios from the same candidate list before the run and wait for graph-element counts to stabilize before deleting the source thread. Episode, edge, and node counts are checked every 30 seconds; stabilization requires three consecutive identical readings. All 12 scenarios meet this condition and have the target detected in thread B's context before deletion. Table 9 reports target detection in context at each checkpoint for both source-thread experiments.

Table 9: Target detection in context after source-thread deletion. A dash indicates a checkpoint not collected in that experiment.

Pre-deletion condition	∼30 s	∼5 min	∼15 min
Target formed and visible in thread B	0/28	28/28	—
Graph-element counts additionally stabilized	0/12	12/12	12/12

In the 12-scenario experiment, the reappearing target preference is found in the user-summary component of context. Both experiments show that early absence from context after source-thread deletion does not imply continued absence at later checkpoints.

C AI Use Statement

We used AI to assist in writing and reviewing code, to help identify recent related work and reduce the risk of overlooking relevant studies, and to polish the language of the manuscript.