UNDERSTANDING AND MITIGATING INFERENCE-TIME OVERRELIANCE USING AGENTIC MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic memory allows LLM agents to reuse past experience, yet retrieved memories can also distort inference even when they are benign, correctly stored, and appropriately retrieved. We study this failure mode, which we call *memory overreliance*. Across benchmarks and memory architectures, we find that memory is useful when past experience transfers to the current task, but can become misleading when only part of the evidence transfers. Failures are strongest under partial query-memory overlap, a pattern further confirmed by controlled experiments that vary the amount of overlapping evidence. Motivated by this finding, we propose MEMTRIM, a plug-and-play framework that indexes memory evidence at write time and controls its reuse at read time. MEMTRIM removes repeated or conflicting evidence while preserving useful memory-specific information, requires no retraining, and applies to both embedding-based and structured memory systems. Experiments show that MEMTRIM reduces memory overreliance while preserving the benefits of useful memory across models and memory settings.

1 INTRODUCTION

Memory allows LLM agents to carry information across interactions and reuse past experience on future tasks. This capability is increasingly important for agents that operate over long horizons, where solving each request in isolation would discard useful experience accumulated from previous interactions (Xiong et al., 2026; Park et al., 2023; Zhong et al., 2024; Shinn et al., 2023; Zhao et al., 2024). However, memory can misguide inference even when the memory itself is benign. A memory may be correctly formed and appropriately retrieved during normal user-agent interactions, yet contain information that does not fully transfer to the current task (Yoran et al., 2024; Yu et al., 2024; Kim & Lee, 2024). Once placed in the inference context, such memory can exert undue influence on the model's reasoning. We refer to this phenomenon as **memory overreliance**: excessive reliance on benign memory that harms inference on the current task.

Our Work. We study this issue in three steps. We first establish when memory overreliance occurs and investigate its cause. We then develop a mitigation method. Finally, we test whether the method reduces overreliance while preserving the benefits of useful memory.

We begin with a three-level empirical analysis of memory overreliance. ❶ First, we ask *whether overreliance arises under ordinary memory use*. Across multiple benchmarks (Cohen et al., 2024; Joishy et al., 2025; Jin et al., 2023) and memory architectures (Chhikara et al., 2025; Xu et al., 2026; Packer et al., 2023; Rasmussen et al., 2025), we find a clear contrast: memory improves inference when the stored interaction remains applicable, but can reduce accuracy when the current task shares substantial evidence with the memory while requiring a different conclusion. ❷ Subsequently, we examine *where these failures occur*. We compare query-memory overlap between successful and failed cases and find a statistically significant difference. Successful cases appear more often near the two extremes of the overlap range, while failures concentrate under partial overlap. In this regime, enough evidence is shared to make the memory influential, yet some answer-relevant information still differs. ❸ Furthermore, we test this relationship directly through controlled experiments that vary the amount of overlapping evidence while holding the remaining memory content fixed. We observe an effect that, with little overlap, memory has limited influence; as overlap increases, its negative effect grows and is strongest at intermediate levels. The effect then reverses as the interaction becomes sufficiently aligned with the stored experience. Collectively, these results indicate that

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.

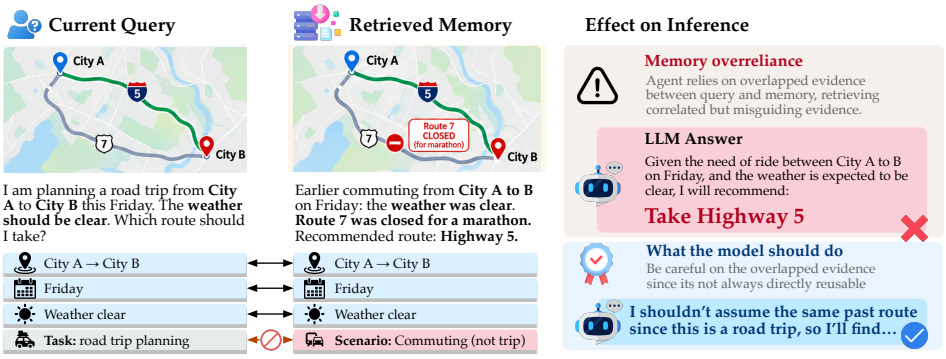

Figure 1: Illustration of memory overreliance. A retrieved memory can appear highly relevant because it shares substantial evidence with the current query, while still carrying context-specific information that does not transfer to the present task. Reusing the memory without separating these pieces of evidence can misguide the model toward an unsupported conclusion.

partial evidence overlap is a key reason in which ordinary memory use can become misleading at inference time.

These findings motivate MEMTRIM[1], a plug-and-play framework that controls how overlapping memory evidence enters inference. At write time, MEMTRIM decomposes each memory into evidence units and indexes them in a trie (Bodon & Rónyai, 2003). Identical evidence maps to shared trie nodes, while different values of the same evidence branch separately. At read time, the current query is mapped into the same evidence space and compared with the retrieved memories. MEMTRIM removes evidence already present in the query, drops conflicting memory evidence, and retains useful information found only in memory. It also removes a previous answer when the evidence it relied on has changed, or preserves it only as contextual evidence when the current query asks a different question. Since MEMTRIM operates only when memory is written and read, it can be added to existing embedding-based and structured memory systems without retraining the underlying LLM.

We evaluate whether this design reduces overreliance without weakening useful memory. Across benchmarks, LLMs, and memory architectures, MEMTRIM reduces memory-induced errors on task-divergent cases while preserving the benefit of memory when the stored interaction remains applicable. Component ablations show that removing repeated and conflicting evidence, handling previous answers when relevant evidence changes, and distinguishing task changes each contribute in the settings they are designed for. We further test MEMTRIM under deliberate memory attacks and find that its performance degrades more gradually as misleading memories increase. Finally, MEMTRIM introduces only small changes in inference time across the evaluated memory systems.

To summarize, our contributions are as follows:

- We identify *memory overreliance* as an inference-time failure of benign agentic memory and show that failures concentrate under partial evidence overlap between the current query and retrieved memory.
- We propose MEMTRIM, a plug-and-play framework that indexes memory at the evidence level and controls how repeated, conflicting, and memory-specific information is reused during inference.
- We evaluate MEMTRIM across benchmarks, LLMs, and memory architectures, including component ablations, memory-attack stress tests, and runtime analysis.

2 AN EMPIRICAL STUDY OF MEMORY OVERRELIANCE

We study memory overreliance through three questions. RQ1 asks whether benign memory can hurt inference under normal memory use. RQ2 examines where these failures occur along the

[1]Memory with *Trie-based Replicate-aware Inference Mitigation*. Code is released at: `https://anonymous.4open.science/r/MEMTRIM-8E5E/`

Table 1: Accuracy (%) on task-aligned (TA) and task-divergent with duplicated evidence (TD-Dup) cases. TA denotes cases where the memory storage is directly applicable, while TD-Dup denotes cases with substantial query-memory overlap but a non-transferable stored conclusion. ✓ →✗ and ✗ →✓ count memory-induced regressions and corrections on TD-Dup cases, respectively. Standard deviations are shown as subscripts.

		RippleEdits				CounterLogic				CLADDER-Derived			
Model	**Memory**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**
Qwen3.8-27B	No Mem.	$68.0_{\pm 0.4}$	$71.3_{\pm 0.8}$	–	–	$85.4_{\pm 1.3}$	$86.1_{\pm 0.9}$	–	–	$60.1_{\pm 0.5}$	$61.2_{\pm 0.9}$	–	–
	Mem0	$85.3_{\pm 0.7}$	$62.6_{\pm 1.3}$	11.6	2.9	$93.1_{\pm 0.9}$	$80.0_{\pm 1.4}$	7.8	1.7	$79.6_{\pm 0.8}$	$46.6_{\pm 1.5}$	16.9	2.3
	A-Mem	$82.6_{\pm 1.0}$	$61.2_{\pm 1.5}$	12.8	2.7	$92.4_{\pm 0.6}$	$80.6_{\pm 1.2}$	7.1	1.6	$78.4_{\pm 0.9}$	$46.1_{\pm 1.4}$	17.6	2.5
	MemGPT	$80.9_{\pm 0.8}$	$65.7_{\pm 1.1}$	8.9	3.3	$91.2_{\pm 1.1}$	$82.8_{\pm 1.0}$	5.2	1.9	$78.1_{\pm 0.7}$	$49.8_{\pm 1.2}$	14.1	2.7
	Graphiti	$87.4_{\pm 0.5}$	$58.3_{\pm 1.7}$	15.4	2.4	$94.6_{\pm 0.7}$	$78.7_{\pm 1.6}$	8.9	1.5	$81.2_{\pm 0.6}$	$43.9_{\pm 1.8}$	19.4	2.1
GPT-5.6 Luna	No Mem.	$65.4_{\pm 0.8}$	$68.9_{\pm 0.7}$	–	–	$84.9_{\pm 1.2}$	$84.7_{\pm 0.8}$	–	–	$58.8_{\pm 0.6}$	$58.4_{\pm 0.8}$	–	–
	Mem0	$86.1_{\pm 0.4}$	$62.2_{\pm 1.2}$	9.7	3.0	$93.4_{\pm 0.7}$	$79.4_{\pm 1.3}$	7.0	1.7	$80.4_{\pm 0.9}$	$45.5_{\pm 1.1}$	15.3	2.4
	A-Mem	$83.7_{\pm 1.0}$	$60.8_{\pm 1.1}$	10.9	2.8	$92.2_{\pm 0.9}$	$80.1_{\pm 1.0}$	6.4	1.8	$78.9_{\pm 1.1}$	$46.2_{\pm 1.6}$	14.8	2.6
	MemGPT	$82.4_{\pm 0.5}$	$65.5_{\pm 0.9}$	6.8	3.4	$91.6_{\pm 0.6}$	$82.1_{\pm 0.8}$	4.6	2.0	$79.2_{\pm 1.0}$	$48.6_{\pm 1.1}$	12.6	2.8
	Graphiti	$88.0_{\pm 0.4}$	$57.8_{\pm 1.5}$	13.6	2.5	$94.3_{\pm 0.5}$	$77.9_{\pm 1.4}$	8.3	1.5	$82.1_{\pm 0.6}$	$43.5_{\pm 1.4}$	17.1	2.2

query-memory overlap range. RQ3 then controls the amount of overlapping evidence to test whether the same pattern persists when the overlap is varied directly.

Benchmarks and Models. We evaluate on RippleEdits (Cohen et al., 2024), CounterLogic (Joishy et al., 2025), and CLADDER-Derived (Jin et al., 2023) using Qwen3.8-27B and GPT-5.6 Luna. For each benchmark, memories are built from earlier interactions with valid stored answers and evaluated on separate queries.

Memory systems. We evaluate four memory systems with different architectures: Mem0 (Chhikara et al., 2025) stores extracted memory entries in a retrieval index; A-Mem (Xu et al., 2026) creates structured notes and dynamically links related memories; MemGPT (Packer et al., 2023) separates active and archival memory; and Graphiti (Rasmussen et al., 2025) represents memory as a relational graph.

Cases and metrics. We divide queries into *task-aligned* (TA) and *task-divergent with overlapping evidence* (TD-Dup) cases. In TA cases, the current query asks the same task as the stored interaction and preserves the answer-relevant evidence, so the retrieved memory is useful. In TD-Dup cases, the query still shares substantial evidence with the stored interaction, but the task or answer-relevant information differs (outdated, conflicting, or irrelevant), so the stored conclusion should not be directly reused. We report task accuracy for both groups. On TD-Dup cases, we additionally track predictions that change from correct to wrong (✓ →✗) and from wrong to correct (✗ →✓) after memory is introduced, separating errors caused by memory from cases where memory improves the prediction.

RQ1: Can ordinary memory use degrade inference?

We first compare each model with and without memory. Because every stored interaction was correct when written, this experiment isolates whether ordinary memory reuse can have negative impacts once the current memory storage no longer supports the same query task.

Observations: Table 1 reveals a clear difference between task-aligned (TA) and task-divergent with overlapping evidence (TD-Dup) cases. In TA cases, retrieved memory is consistently useful across datasets and memory architectures because the stored interaction remains applicable to the current task. In TD-Dup cases, the same memory can instead reduce accuracy. At the example level, memory-induced regressions occur much more often than memory-induced corrections.

Insights: This contrast suggests that the problem is not that memory is generally noisy or unhelpful. The same memory mechanism benefits inference when the stored experience transfers to the current task, yet becomes misguiding when only part of that experience transfers. Memory overreliance therefore arises from how partially overlapping evidence is reused during inference.

RQ2: Where does memory overreliance occur?

We next examine which TD-Dup cases are the most vulnerable. We call a case *successful* when it remains correct after memory is added and *failed* when memory changes a correct prediction into an incorrect one. For each case, we compute the content-word Jaccard overlap between the current

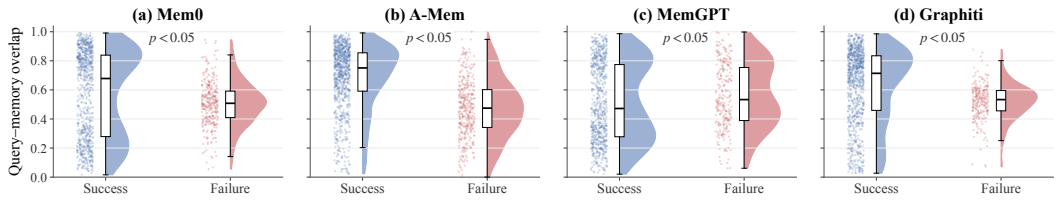

Figure 2: Query-memory overlap (Jaccard similarity) for successful and failed TD-Dup cases on RippleEdits. The statistical test (Simonsohn, 2018) assesses whether the overlap distributions differ between the two groups.

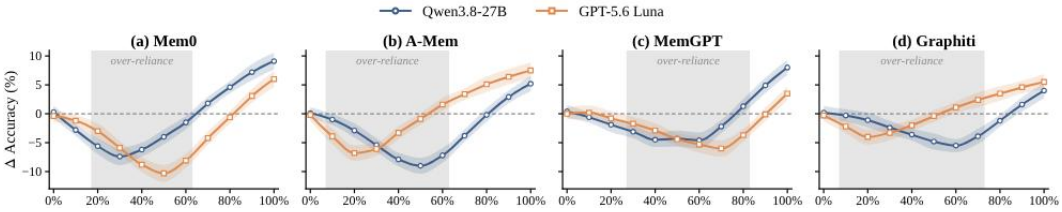

Figure 3: Change in accuracy as the amount of overlapping evidence is varied from task-divergent toward task-aligned interaction. The y-axis reports the accuracy change relative to inference without memory. The gray region marks the range where memory is most misguiding on average.

query and retrieved memory, excluding the stored answer. We refer to this quantity as *query-memory overlap* and compare its distribution between the two groups.

Observations: Figure 2 shows that successful and failed cases occupy different parts of the overlap range. Successful cases occur more often near the two extremes: very little overlap gives memory limited influence, while very high overlap indicates that the current task closely resembles the stored interaction. Failed cases instead concentrate under partial overlap, with a statistically significant separation across all four memory systems.

Insights: This pattern suggests that overlap plays two roles. Increasing overlap makes retrieved memory more influential but does not guaranty that the evidence supporting its conclusion also transfers to the current task. Partial overlap is therefore particularly vulnerable: enough evidence is overlapped to make the memory dominant, while some answer-relevant information can still diverge. The exact distribution varies across memory architectures, but this pattern remains consistent. More results are provided in Appendix B.

RQ3: How does overreliance change with overlapping evidence?

RQ2 examines TD-Dup cases as they naturally occur, where factors beyond overlap may also distinguish successful from failed cases. To isolate the role of overlap, we construct a controlled setting in which each memory is decomposed into evidence units and vary the fraction repeated in the current query while keeping the remaining content fixed. As overlap increases, the answer-relevant evidence is gradually restored and the interaction moves toward the TA setting. At each level, we measure the memory effect as $\Delta\mathrm{Acc} = \mathrm{Acc}_{\mathrm{mem}} - \mathrm{Acc}_{\text{no-mem}}$ and test the non-monotonic trend with a quadratic term and the two-lines test (Simonsohn, 2018).

Observations: Figure 3 shows a consistent non-monotonic trend across models and memory systems. With little overlapping evidence, memory has limited influence. As overlap increases, its harmful effect grows and is strongest at intermediate overlap. The effect then reverses as the current query becomes more aligned with the stored interaction.

Insights: This observation further explains RQ2. overlap first increases memory influence without making the stored conclusion valid for the current task. At intermediate overlap, substantial evidence is repeated while some decisive information still differs. As the interaction approaches TA, that answer-relevant evidence is restored and the stored conclusion becomes reusable. Memory overreliance therefore emerges when evidence overlap is high enough to amplify memory, but task alignment is still incomplete.

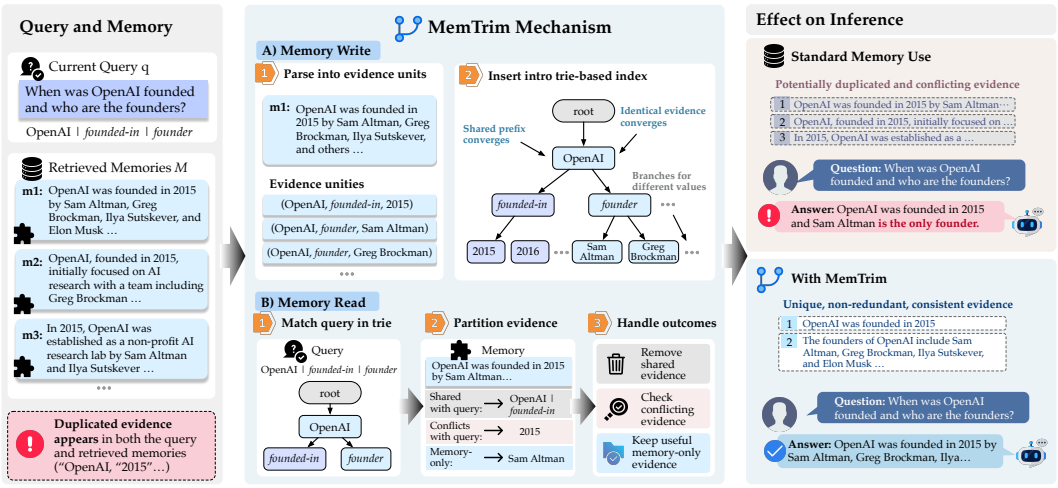

Figure 4: Overview of MEMTRIM. At write time, memories are decomposed into evidence units and linked through a trie-backed index. At read time, MEMTRIM identifies shared, conflicting, and memory-only evidence, then constructs a non-redundant memory context before inference.

Overall Insight. RQ1-RQ3 consistently deliver an empirical insight: memory is useful in task-aligned cases, but can be misguiding **when overlapping evidence is substantial but not fully aligned with the task**, which motivates our subsequent mitigation strategy.

3 METHOD

Our empirical study shows that memory overreliance is most influential when a retrieved memory partially overlaps with the current query. This motivates a simple mitigation principle: **explicitly identify replicated evidence and control how it enters inference.** We develop MEMTRIM to implement this principle. Figure 4 gives an overview. We next describe the evidence representation (§3.1), write-time indexing (§3.2), and read-time memory use (§3.3).

3.1 OVERVIEW AND EVIDENCE REPRESENTATION

MEMTRIM is a plug-and-play layer around an existing memory system. The underlying system continues to store complete memories and retrieve relevant entries using its original mechanism. MEMTRIM adds an evidence index when a memory is written and processes retrieved memories before they are passed to the LLM. It therefore requires neither model retraining nor replacement of the original memory architecture.

Consider a stored interaction $m = (x_m, y_m)$, where x_m contains the query and its context and y_m is the resulting answer. We parse x_m into a set of atomic evidence units E_m. Each unit is represented as a key-value pair $e = (k, v)$, where k identifies an attribute or relation and v gives its value. We further extract a task signature τ_m that specifies what the interaction asks the model to determine. The stored answer is represented as a typed outcome $o_m = (\tau_m, y_m)$ and associated with a support set $H_m \subseteq E_m$, which contains the evidence on which that outcome depends. Thus, a memory is represented as $m = (E_m, \tau_m, H_m, o_m)$. The same parser is applied to a current query q, producing E_q and τ_q. For free-text memory, we obtain these fields with a fixed instruction-based parser. If the underlying memory already contains structured metadata (e.g., graph-based memory (Rasmussen et al., 2025)), they can be mapped directly into the same representation. Details of implementations (e.g., evidence parsing and alignment) are provided in Appendix A.1.

Illustrative Example

Consider a stored interaction: *"Northstar Robotics was founded in 2016 by Maya Lee and Daniel Ortiz. When was Northstar Robotics founded?"* The stored answer is *"2016."*

MEMTRIM parses the interaction into evidence units such as $(\texttt{entity}, \texttt{NorthstarRobotics})$, $(\texttt{founded_in}, 2016)$, $(\texttt{founder}, \texttt{MayaLee})$, and $(\texttt{founder}, \texttt{DanielOrtiz})$. Its task signature is $\tau_m = \texttt{founding_year}$, while the stored outcome is $o_m = (\texttt{founding_year}, 2016)$. Since the answer depends directly on the founding-year evidence, $(\texttt{founded_in}, 2016) \in H_m$.

3.2 MEMORY WRITE: TRIE-BACKED EVIDENCE INDEX

At write time, MEMTRIM inserts each evidence unit into a global trie $\mathcal{T}$. Evidence units are indexed independently, so one memory can be linked to multiple trie nodes. For an evidence unit $e = (k, v)$, the trie first follows its canonical key k and then its value v. The corresponding node records the memories that contain this evidence and whether the unit belongs to their support sets H_m. Complete memories remain in the original memory store; MEMTRIM maintains only this evidence-level index.

The trie makes shared and changed evidence directly distinguishable. Memories containing the same key-value unit point to the same node, whereas different values of the same key occupy different branches. For example, `founded_in=2016` and `founded_in=2017` share the same relation key but diverge at the value level.

Sparse evidence overlap. Shared evidence need not form a continuous prefix between a memory and a query. Suppose $E_m = \{e_1, e_2, e_3, e_4\}$ and $E_q = \{e_1, e_2', e_3, e_4'\}$. Because each unit is indexed independently, MEMTRIM can still match e_1 and e_3 even though the intervening units differ. This allows the index to capture replicated evidence at arbitrary positions within an interaction.

3.3 MEMORY READ: REPLICATE-AWARE CONTEXT CONSTRUCTION

At inference time, the underlying memory system first retrieves its usual candidate set $\mathcal{R}(q)$. MEMTRIM parses the current query into E_q and looks up its evidence units independently in the trie. For each retrieved memory m, the resulting matches divide its evidence into three sets: (i) *shared evidence* $S_m(q)$, whose key and value also appear in the query; (ii) *conflicting evidence* $D_m(q)$, whose key appears in the query with a different value; and (iii) *memory-only evidence* $U_m(q)$, which does not appear in the query. Since matching is performed independently for each unit, shared evidence can occur anywhere in the memory.

MEMTRIM then constructs the memory context given to the LLM. Shared evidence is removed from the retrieved memory because it is already stated in the query, avoiding repeated support for the same information. Conflicting evidence is also removed because the current query supplies the value for the current inference. Memory-only evidence is retained when it provides additional information absent from the query.

The stored outcome o_m is handled separately. If any supporting evidence conflicts with the current query, i.e., $D_m(q) \cap H_m \neq \emptyset$, MEMTRIM suppresses the previous outcome. Otherwise, its use depends on the task. When $\tau_m = \tau_q$, the outcome can be retained as a previous answer to the same task. When $\tau_m \neq \tau_q$, MEMTRIM converts it into an evidence statement labeled by its original task. For example, a previous answer "2016" to a founding-year query can be represented as $(\texttt{founded_in}, 2016)$ when a later query asks who founded the same organization. The earlier result can then remain available as contextual evidence without being presented as the answer to the new question. The transformed context from one memory is

$$\widetilde{m}(q) = U_m(q) \cup \begin{cases} \{o_m\}, & D_m(q) \cap H_m = \emptyset,\ \tau_m = \tau_q, \\ \{\mathrm{Type}(o_m)\}, & D_m(q) \cap H_m = \emptyset,\ \tau_m \neq \tau_q, \\ \emptyset, & D_m(q) \cap H_m \neq \emptyset, \end{cases} \tag{1}$$

where $\mathrm{Type}(o_m)$ converts the previous outcome into an evidence unit labeled by its original task. Outcomes that cannot be meaningfully reused as evidence are omitted.

Finally, MEMTRIM merges the transformed candidate memories and emits each retained evidence unit only once. This context is combined with the unchanged current query and passed to the LLM.

Table 2: Mitigation results (%). All methods use Mem0 as the memory backbone and differ only in how retrieved memory is filtered, compressed, or used. TA denotes task-aligned cases where the stored memory is directly applicable, and TD-Dup denotes task-divergent cases with duplicated evidence, where the stored conclusion does not transfer. ✓ →✗ and ✗ →✓ count memory-induced regressions and corrections on TD-Dup cases, respectively. Standard deviations are shown as subscripts.

		RippleEdits				CounterLogic				CLADDER-Derived			
Model	**Method**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**
Qwen3.8-27B	No Mem.	$68.0_{\pm 0.4}$	$71.3_{\pm 0.8}$	–	–	$85.4_{\pm 1.3}$	$86.1_{\pm 0.9}$	–	–	$60.1_{\pm 0.5}$	$61.2_{\pm 0.9}$	–	–
	Mem0	$85.3_{\pm 0.7}$	$62.6_{\pm 1.3}$	11.6	2.9	$93.1_{\pm 0.9}$	$80.0_{\pm 1.4}$	7.8	1.7	$79.6_{\pm 0.8}$	$46.6_{\pm 1.5}$	16.9	2.3
	Facts-only	$71.6_{\pm 0.6}$	$72.4_{\pm 0.9}$	2.2	3.3	$87.0_{\pm 1.3}$	$86.7_{\pm 0.7}$	1.5	2.1	$64.1_{\pm 1.2}$	$61.0_{\pm 1.1}$	3.2	3.0
	Top-1	$74.5_{\pm 1.2}$	$63.1_{\pm 1.5}$	10.8	2.6	$88.3_{\pm 0.5}$	$79.4_{\pm 1.2}$	8.3	1.6	$67.4_{\pm 0.6}$	$47.6_{\pm 0.7}$	15.7	2.1
	MMR	$85.7_{\pm 1.6}$	$68.9_{\pm 1.1}$	5.4	3.0	$93.3_{\pm 0.7}$	$84.3_{\pm 0.9}$	3.6	1.8	$80.2_{\pm 1.3}$	$55.7_{\pm 1.9}$	7.9	2.4
	Sim. filter	$75.6_{\pm 0.9}$	$66.4_{\pm 1.8}$	7.9	3.0	$88.8_{\pm 1.8}$	$82.6_{\pm 0.7}$	5.3	1.8	$68.7_{\pm 1.6}$	$52.1_{\pm 1.0}$	11.5	2.4
	RECOMP	$84.1_{\pm 1.4}$	$63.1_{\pm 1.4}$	11.0	2.8	$92.6_{\pm 0.9}$	$80.4_{\pm 1.7}$	7.4	1.7	$78.2_{\pm 0.7}$	$47.4_{\pm 1.4}$	16.0	2.2
	LLMLingua-2	$79.6_{\pm 0.6}$	$60.7_{\pm 1.2}$	13.3	2.7	$90.6_{\pm 1.2}$	$78.8_{\pm 0.7}$	8.9	1.6	$73.5_{\pm 0.6}$	$44.0_{\pm 0.9}$	19.4	2.2
	Context-faithful	$86.2_{\pm 0.5}$	$72.6_{\pm 1.4}$	1.4	2.7	$93.5_{\pm 0.9}$	$86.8_{\pm 1.4}$	0.9	1.6	$82.8_{\pm 1.1}$	$61.3_{\pm 1.0}$	2.0	2.1
	Answer-first	$77.9_{\pm 1.5}$	$71.2_{\pm 0.6}$	3.5	3.4	$89.8_{\pm 0.8}$	$85.6_{\pm 1.4}$	2.4	1.9	$71.3_{\pm 1.2}$	$58.9_{\pm 1.8}$	5.1	2.8
	Micro-Act	$82.9_{\pm 0.6}$	$69.8_{\pm 2.3}$	5.0	3.5	$92.0_{\pm 1.8}$	$84.7_{\pm 0.8}$	3.4	2.0	$76.4_{\pm 1.0}$	$56.8_{\pm 1.7}$	7.3	2.9
	MEMTRIM (ours)	**$88.4_{\pm 1.7}$**	**$74.5_{\pm 1.3}$**	**0.6**	**3.8**	**$94.6_{\pm 0.6}$**	**$87.9_{\pm 1.5}$**	**0.4**	**2.2**	**$83.9_{\pm 1.5}$**	**$63.5_{\pm 1.4}$**	**0.9**	**3.2**
GPT-5.6 Luna	No Mem.	$65.4_{\pm 0.9}$	$68.9_{\pm 0.7}$	–	–	$84.9_{\pm 1.2}$	$84.7_{\pm 1.8}$	–	–	$58.8_{\pm 0.6}$	$58.4_{\pm 1.4}$	–	–
	Mem0	$86.1_{\pm 0.4}$	$62.2_{\pm 1.2}$	9.7	3.0	$93.4_{\pm 0.7}$	$79.4_{\pm 1.3}$	7.0	1.7	$80.4_{\pm 0.9}$	$45.5_{\pm 1.1}$	15.3	2.4
	Facts-only	$70.0_{\pm 0.9}$	$72.6_{\pm 1.6}$	0.9	4.6	$85.7_{\pm 1.3}$	$85.5_{\pm 1.2}$	1.3	2.1	$63.5_{\pm 1.7}$	$58.4_{\pm 1.3}$	3.1	3.1
	Top-1	$73.4_{\pm 0.6}$	$62.3_{\pm 1.6}$	9.3	2.7	$88.2_{\pm 1.8}$	$79.6_{\pm 1.8}$	6.7	1.6	$66.6_{\pm 1.0}$	$46.8_{\pm 1.5}$	13.8	2.2
	MMR	$86.4_{\pm 1.1}$	$67.8_{\pm 0.8}$	4.2	3.1	$93.6_{\pm 0.6}$	$83.4_{\pm 1.7}$	3.1	1.8	$80.9_{\pm 0.8}$	$54.2_{\pm 1.1}$	6.7	2.5
	Sim. filter	$75.0_{\pm 0.6}$	$65.0_{\pm 1.2}$	7.0	3.1	$89.7_{\pm 1.6}$	$81.7_{\pm 1.7}$	4.8	1.8	$68.8_{\pm 0.9}$	$49.9_{\pm 1.2}$	11.0	2.5
	RECOMP	$84.2_{\pm 1.6}$	$62.8_{\pm 1.9}$	9.0	2.9	$92.4_{\pm 0.7}$	$80.1_{\pm 0.9}$	6.3	1.7	$76.0_{\pm 1.1}$	$46.8_{\pm 1.4}$	13.9	2.3
	LLMLingua-2	$78.8_{\pm 0.5}$	$61.0_{\pm 1.2}$	10.7	2.8	$90.5_{\pm 1.2}$	$78.4_{\pm 1.9}$	7.9	1.6	$73.7_{\pm 1.2}$	$42.6_{\pm 1.5}$	18.1	2.3
	Context-faithful	$87.0_{\pm 0.6}$	$70.5_{\pm 1.9}$	1.2	2.8	$93.8_{\pm 1.6}$	$85.4_{\pm 1.7}$	0.9	1.6	$81.6_{\pm 1.0}$	$58.8_{\pm 0.7}$	1.8	2.2
	Answer-first	$77.5_{\pm 0.6}$	$69.4_{\pm 0.7}$	3.0	3.5	$86.5_{\pm 0.7}$	$84.6_{\pm 1.1}$	2.0	1.9	$70.3_{\pm 0.5}$	$61.8_{\pm 0.8}$	1.4	4.8
	Micro-Act	$82.1_{\pm 1.0}$	$68.6_{\pm 0.6}$	3.9	3.6	$91.7_{\pm 1.3}$	$83.5_{\pm 0.8}$	3.2	2.0	$79.7_{\pm 1.0}$	$55.1_{\pm 1.1}$	6.3	3.0
	MEMTRIM (ours)	**$89.0_{\pm 0.7}$**	**$75.3_{\pm 1.8}$**	**0.5**	**6.9**	**$94.8_{\pm 1.8}$**	**$88.5_{\pm 1.3}$**	**0.4**	**4.2**	**$84.5_{\pm 1.1}$**	**$64.9_{\pm 0.7}$**	**0.8**	**7.3**

4 EVALUATION

We evaluate whether MEMTRIM reduces memory overreliance while preserving the benefit of useful memory. We detail settings and evaluations as below:

Evaluation settings. To make MEMTRIM directly comparable with the previous results. We use the same benchmarks, backbone LLMs, memory systems, and metrics as in §2.

Baselines: We compare against several representative strategies applied on top of each agentic memory system: (i) *Facts-only* removes the stored answer and retains only factual content; (ii) *Top-1* keeps only the highest-ranked retrieved memory; (iii) MMR (Carbonell & Goldstein, 1998) reranks memories to reduce redundancy; (iv) *Similarity Filter* removes memories below a similarity threshold; (v) RECOMP (Xu et al., 2024) compresses retrieved memories into task-relevant evidence; (vi) LLMLingua-2 (Pan et al., 2024) shortens the retrieved context through prompt compression; (vii) *Context-faithful* (Zhou et al., 2023) prompts the model to prioritize the current context under retrieval conflict; (viii) *Answer-first* (Neeman et al., 2023) answers from the current query before comparing against retrieved information; and (ix) Micro-Act (Huo et al., 2025) adds an explicit reasoning step for handling conflicting retrieved knowledge.

4.1 DOES MEMTRIM MITIGATE OVERRELIANCE WHILE PRESERVING USEFUL MEMORY?

We first compare all methods using the same underlying retrieved memories. The main question is whether a method can improve reasoning on TD-Dup cases while preserving the benefits of memory on TA cases. Table 2 shows that MEMTRIM improves this trade-off. On TD-Dup cases, it recovers much of the performance lost under unmodified memory, while TA performance remains close to (or even better than) the original memory system. The same trend appears for different LLM backbones and extends to the other memory architectures in the Appendix B.3. Hence, we summarize that MEMTRIM reduces incorrect memory-based inference while retaining (or improving) cases where memory is helpful given the structured representation in MEMTRIM.

The comparison also clarifies why simply weakening memory is insufficient. Baselines that remove entries, discard stored answers, or compress the retrieved context can reduce harmful influence

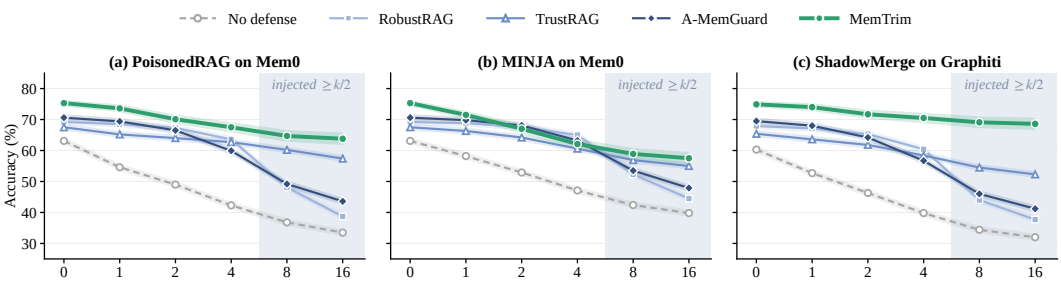

Figure 6: TD-Dup accuracy as misleading memory is increasingly injected into the retrieved context. The shaded region marks settings where injected entries form a large fraction of retrieval. Bands show confidence intervals across runs.

on TD-Dup cases, but can also discard information that remains useful in TA cases. Prompting-based methods preserve more of the original context, yet overlapping evidence can still reinforce a conclusion that does not transfer to the current task. MEMTRIM works at the evidence level, controlling replicated or conflicting memory evidence while retaining memory-specific information that remains useful.

4.2 WHICH PARTS OF MEMTRIM ARE RESPONSIBLE FOR THE MITIGATION?

We ablate the main read-time steps in MEMTRIM. *Keep shared* retains evidence already present in the current query; *Keep conflicts* retains memory evidence that conflicts with the query; *Keep old answer* keeps the previous answer even when the evidence it relied on has changed; *Ignore task change* directly reuses the previous answer when the current query asks a different question; and *Embedding match* replaces trie-based evidence matching with embedding similarity.

Figure 5 shows that these steps matter in different settings. Keeping shared evidence consistently lowers TD-Dup accuracy, showing that repeated evidence contributes to overreliance. Keeping conflicting evidence or the old answer has a larger effect on RippleEdits and CLADDER-Derived, where answer-relevant information changes. Ignoring the task change matters most on CounterLogic, where much of the evidence stays the same but the question changes. Embedding matching causes a smaller but consistent drop.

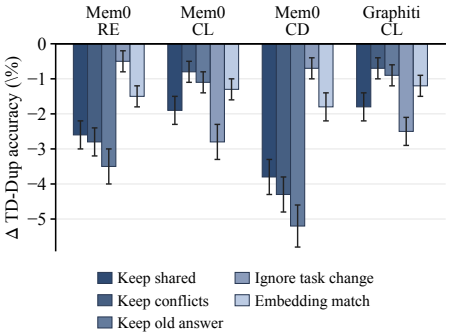

Figure 5: Ablation of MEMTRIM on TD-Dup cases with Qwen3.8-27B. Bars show the accuracy change relative to the full method; lower values indicate a larger drop. RE, CL, and CD denote RippleEdits, CounterLogic, and CLADDER-Derived.

Overall, the ablations follow the design of MEMTRIM: removing repeated evidence reduces duplicated influence, removing conflicts keeps memory consistent with the current query, dropping an outdated answer prevents it from carrying over after its evidence changes, and distinguishing task changes prevents an answer to one question from being reused for another.

4.3 DOES MEMTRIM REMAIN EFFECTIVE WHEN MEMORY IS ATTACKED?

Our main experiments use memories formed through ordinary interaction. We further test MEMTRIM under deliberate memory attacks. PoisonedRAG (Zou et al., 2025) injects repeated stale records, MINJA (Dong et al., 2026) induces the agent to store such records through normal queries, and ShadowMerge (Luo et al., 2026) inserts conflicting relations into graph memory. We increase the number of injected entries while keeping the retrieval budget fixed, and compare MEMTRIM with RobustRAG (Xiang et al., 2024), TrustRAG (Zhou et al., 2025), and A-MemGuard (Wei et al., 2025).

Figure 6 shows that all methods degrade as attacked memories occupy more of the retrieved context, but MEMTRIM degrades more slowly. The difference becomes clearer as injected entries make up a

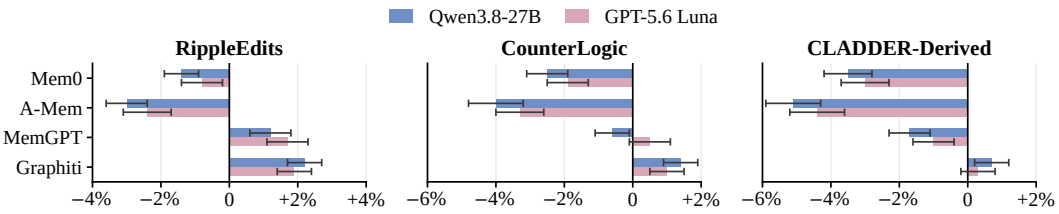

Figure 7: Change in total reasoning time after adding MEMTRIM to each memory system. Negative values indicate shorter reasoning time. Error bars show variation across runs.

larger share of the retrieved memories, and the same pattern appears across both text-based and graph-based attacks. The result highlights a useful property of MEMTRIM: repeated stale memories do not translate directly into repeated influence on the model. MEMTRIM removes duplicated evidence and excludes conflicting information before inference, so injecting more copies has a weaker effect on the final context. This suggests that the same design used to reduce benign memory overreliance also provides some resistance when misleading memory is deliberately amplified.

4.4 DOES MEMTRIM INTRODUCE SIGNIFICANT RUNNING OVERHEAD?

We measure total reasoning time after adding MEMTRIM, including memory reading and answer generation; stored memories are processed once at write time and reused across queries. Figure 7 shows only small runtime changes across memory systems. MEMTRIM adds some cost when matching evidence, but it can also shorten the retrieved context before generation, which offsets part of this cost and can even reduce runtime in some settings. Overall, MEMTRIM improves memory use without substantially increasing inference time.

5 RELATED WORK

Memory in LLM agents. Memory systems let an agent keep and reuse past experience. One line of works build the storage: long-term memory across sessions (Park et al., 2023; Zhong et al., 2024), context hierarchies (Packer et al., 2023), associative indexes (Gutiérrez et al., 2024), and self-organized memory (Xu et al., 2026). Other works turn past trajectories into reusable knowledge such as reflections, lessons, and workflows (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024), and recent benchmarks evaluate these systems in multi-turn interaction (Hu et al., 2026). All of these designs leverage retrieved relevant memory to help the current reasoning tasks.

Failures and attacks on memory reuse. Memory can also hurt. Agents copy retrieved experiences with similar inputs, so past errors spread (Xiong et al., 2026). Memories are applied where they no longer fit (Pulipaka et al., 2026), turn faulty under repeated updates (Zhang et al., 2026a), or look similar to the query while being wrong for the task (Zhang et al., 2026b). Memory is also an attack surface: adversaries poison memory banks (Chen et al., 2024), retrieval corpora (Zou et al., 2025), or self-evolving experiences (Wang et al., 2026), and defenses curate data against such samples (Liu et al., 2024). In all these cases, the failure comes from a flawed memory, a flawed retrieval, or an attacker. However, memory overreliance needs none of them: a correct, well-retrieved memory fails simply because its answer-relevant condition has changed.

6 CONCLUSION

We study memory overreliance, where benign and appropriately retrieved memory can misguide inference when only part of the stored experience transfers to the current task. Our experiments identify partial evidence overlap as the most vulnerable regime and motivate MEMTRIM, which controls repeated and conflicting memory evidence before inference while preserving useful memory-specific information. Across models and memory architectures, MEMTRIM reduces harmful memory reuse while retaining the benefits of applicable memory, with small runtime changes and added resistance under memory attacks. **Ethics Statement:** This work uses existing research benchmarks, involves no human subjects or collection of personal or sensitive data, and is intended to improve the reliability of memory-augmented LLM agents.

AI USE STATEMENT

In this work, we used generative AI tools only for language polishing, grammar correction, and improving the readability of author-written text. We did not use generative AI tools to generate synthetic datasets; develop theoretical models or conceptual frameworks; formulate mathematical claims; provide critical ingredients for proving mathematical claims; assist in writing proofs; propose or refine hypotheses; design or provide feedback on research methodology or experiments; implement methods; assist with translation; clean or reformat datasets; support qualitative or thematic data analysis; or interpret results. We reviewed all AI-assisted edits and take responsibility for the final content of this work, including all text, claims, and artifacts.

REPRODUCIBILITY STATEMENT

We provide an anonymized implementation of MEMTRIM at `https://anonymous.4open.science/r/MEMTRIM-8E5E/`. Section 3 specifies the evidence representation, write-time indexing, and read-time memory processing, while Section 4 describes the benchmarks, models, memory systems, baselines, and metrics. Appendix A.1 gives implementation details for evidence parsing and alignment, and Appendix C provides the deterministic construction of TA and TD-Dup evaluation pairs. Together with the released code, these details support reproduction of the reported experiments.

REFERENCES

Ferenc Bodon and Lajos Rónyai. Trie: an alternative data structure for data mining algorithms. *Mathematical and Computer Modelling*, 38(7-9):739–751, 2003.

Jaime G Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, volume 98, pp. 290941–291025, 1998.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *Advances in Neural Information Processing Systems*, 37:130185–130213, 2024.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298, 2024.

Shen Dong, Shaochen Xu, Pengfei He, Yige Li, Jiliang Tang, Tianming Liu, Hui Liu, and Zhen Xiang. Memory injection attacks on llm agents via query-only interaction. *Advances in Neural Information Processing Systems*, 38:46697–46731, 2026.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

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.

Nan Huo, Jinyang Li, Bowen Qin, Ge Qu, Xiaolong Li, Xiaodong Li, Chenhao Ma, and Reynold Cheng. Micro-act: Mitigating knowledge conflict in llm-based rag via actionable self-reasoning. *arXiv preprint arXiv:2506.05278*, 2025.

Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng Lyu, Kevin Blin, Fernando Gonzalez Adauto, Max Kleiman-Weiner, Mrinmaya Sachan, et al. Cladder: Assessing causal reasoning in language models. *Advances in Neural Information Processing Systems*, 36:31038–31065, 2023.

Anish R Joishy, Ishwar B Balappanawar, Vamshi Krishna Bonagiri, Manas Gaur, Krishnaprasad Thirunarayan, and Ponnurangam Kumaraguru. Flying pigs, far and beyond: Evaluating llm reasoning in counterfactual worlds. *arXiv preprint arXiv:2505.22318*, 2025.

Kiseung Kim and Jay-Yoon Lee. Re-rag: Improving open-domain qa performance and interpretability with relevance estimator in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22149–22161, 2024.

Xiaoqun Liu, Jiacheng Liang, Luoxi Tang, Muchao Ye, Weicheng Ma, and Zhaohan Xi. Data to defense: The role of curation in customizing llms against jailbreaking attacks. *arXiv preprint arXiv:2410.02220*, 2024.

Yang Luo, Zifeng Kang, Tiantian Ji, Xinran Liu, Yong Liu, Shuyu Li, and Lingyun Peng. Shadowmerge: A novel poisoning attack on graph-based agent memory via relation-channel conflicts. *arXiv preprint arXiv:2605.09033*, 2026.

Ella Neeman, Roee Aharoni, Or Honovich, Leshem Choshen, Idan Szpektor, and Omri Abend. Disentqa: Disentangling parametric and contextual knowledge with counterfactual question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10056–10070, 2023.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024.

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*, pp. 1–22, 2023.

Sidharth Pulipaka, Oliver Chen, Manas Sharma, Taaha S Bajwa, Vyas Raina, and Ivaxi Sheth. Persistbench: When should long-term memories be forgotten by llms? *arXiv preprint arXiv:2602.01146*, 2026.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Uri Simonsohn. Two lines: A valid alternative to the invalid testing of u-shaped relationships with quadratic regressions. *Advances in Methods and Practices in Psychological Science*, 1(4):538–555, 2018.

Kaixiang Wang, Jiong Lou, Zhaojiacheng Zhou, and Jie Li. Oep: Poisoning self-evolving llm agents via locally correct but non-transferable experiences. *arXiv preprint arXiv:2605.18930*, 2026.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Qianshan Wei, Tengchao Yang, Yaochen Wang, Xinfeng Li, Lijun Li, Zhenfei Yin, Yi Zhan, Thorsten Holz, Zhiqiang Lin, and XiaoFeng Wang. A-memguard: A proactive defense framework for llm-based agent memory. *arXiv preprint arXiv:2510.02373*, 2025.

Chong Xiang, Tong Wu, Zexuan Zhong, David Wagner, Danqi Chen, and Prateek Mittal. Certifiably robust rag against retrieval corruption. *arXiv preprint arXiv:2405.15556*, 2024.

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/.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making retrieval-augmented language models robust to irrelevant context. In *International Conference on Learning Representations*, volume 2024, pp. 29862–29883, 2024.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Peixin Cao, Kaixin Ma, Jian Li, Hongwei Wang, and Dong Yu. Chain-of-note: Enhancing robustness in retrieval-augmented language models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 14672–14685, 2024.

Dylan Zhang, Yanshan Lin, Zhengkun Wu, Yihang Sun, Bingxuan Li, Dianqi Li, and Hao Peng. Useful memories become faulty when continuously updated by llms. *arXiv preprint arXiv:2605.12978*, 2026a.

Jiawen Zhang, Kejia Chen, Jiachen Ma, Yangfan Hu, Lipeng He, Yechao Zhang, Jian Liu, Xiaohu Yang, Tianwei Zhang, and Ruoxi Jia. Beyond similarity: Trustworthy memory search for personal ai agents. *arXiv preprint arXiv:2606.06054*, 2026b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

Huichi Zhou, Kin-Hei Lee, Zhonghao Zhan, Yue Chen, Zhenhao Li, Zhaoyang Wang, Hamed Haddadi, and Emine Yilmaz. Trustrag: Enhancing robustness and trustworthiness in retrieval-augmented generation. *arXiv preprint arXiv:2501.00879*, 2025.

Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. Context-faithful prompting for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14544–14556, 2023.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. {PoisonedRAG}: Knowledge corruption attacks to {Retrieval-Augmented} generation of large language models. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 3827–3844, 2025.

A Complementary Technical Details of MemTrim

A.1 Implementation Details

This part provides implementation details omitted from §3. We focus on how free-text interactions are converted into stable evidence units, how answer-support sets and task signatures are constructed, and how evidence from multiple retrieved memories is combined at read time.

Canonicalizing evidence units. The same fact can be expressed in many ways, so MEMTRIM does not index raw text directly. For each interaction, we use a fixed parser prompt to extract typed evidence records. We retain the notation $e = (k, v)$ from §3.1, but internally the key k contains the entity being described, its attribute or relation, and its value type. For example, "Highway 5 is closed," "Route 5 has been shut," and "Highway 5 is not open" can all map to $(\texttt{Highway5_status}, \texttt{closed})$. In contrast, the word "open" in "the restaurant is open" is assigned a different key because it refers to a different entity and attribute. This prevents surface similarity alone from merging evidence with different meanings.

When a new unit is parsed, MEMTRIM first checks for an exact canonical key in the trie. If none exists, it retrieves a small set of existing keys with the same entity and value type and asks the parser whether the new key is semantically equivalent to any of them. Equivalent keys are merged; otherwise a new key is created. Values are aligned only after their keys match. Boolean, categorical, numeric, and temporal values are normalized into a fixed form; free-text values use the same equivalence check. When equivalence is unclear, MEMTRIM keeps the units separate. This conservative rule avoids merging two pieces of evidence only because they share similar wording.

Each parsed unit also keeps a pointer to the text span from which it was extracted. We use this span to verify that every indexed unit is grounded in the original interaction. A parsed unit that cannot be traced to the input is discarded before insertion into the trie.

Canonical Evidence Alignment

Suppose one memory states *"Highway 5 is closed today"*, while another states *"Route 5 has been shut for the day."* Both are normalized to $(\texttt{Highway5_status}, \texttt{closed})$ and therefore share one trie node. If a third memory states *"Highway 5 is open"*, it shares the same key but follows a different value branch, `Highway5_status=open`.
The word *open* may also occur in an unrelated sentence such as *"the visitor center is open."* This evidence is indexed under a different key, e.g., $(\texttt{VisitorCenter_status}, \texttt{open})$, and is never merged with the highway-status evidence.

Extracting task signatures and answer support. The parser also produces the task signature τ_m and support set H_m. A task signature records both the requested operation and its target. For example, "Which route should I take?" and "Recommend a route to City B" map to the same `route_planning` signature, whereas "How long will the trip take?" maps to `travel_time`. We canonicalize task signatures with the same procedure used for evidence keys. If two task descriptions cannot be confidently aligned, we treat them as different tasks; this prevents an earlier answer from being reused under an uncertain task match.

To construct H_m, the parser marks which evidence units directly support the stored outcome o_m. For the route-planning example, the answer "Take Highway 5" may depend on $(\texttt{Highway5_status}, \texttt{open})$ and $(\texttt{destination}, B)$, while an unrelated fact such as the weather need not belong to H_m. We use a conservative policy for uncertain cases: evidence is included in H_m when the parser cannot determine whether the stored outcome depends on it. This can suppress reuse more often, but avoids keeping an outcome after potentially important support has changed.

When $\tau_m \neq \tau_q$, MEMTRIM converts a previous outcome into evidence only when it can be expressed as a factual key-value unit without changing its meaning. For example, "Take Highway 5" can be converted to $(\texttt{recommended_route}, \texttt{Highway5})$ when a later query asks about travel time. Outputs that are inherently procedural, subjective, or cannot be represented faithfully as evidence are omitted under a task change.

Combining multiple retrieved memories. The trie is used only to process memories already returned by the underlying retriever. Given $\mathcal{R}(q)$, MEMTRIM restricts all trie matches to the memory identifiers in this set; it does not use the trie as an additional retrieval system.

After each retrieved memory is transformed as described in §3.3, MEMTRIM merges the retained evidence using its canonical trie identifiers. Identical memory-only evidence from several memories is emitted once. If several retrieved memories provide different values for the same key and the current query does not resolve the conflict, MEMTRIM does not select a value based only on retrieval similarity. By default, all unresolved values for that key are withheld from the final context. If the base memory system provides an explicit ordering signal such as a trusted timestamp, that metadata can be used to resolve the conflict before this step.

The current query always takes precedence when it provides a value for the same key. Thus, if the stored memory contains `Highway5_status=open` while the current query contains `Highway5_status=closed`, the stored value is removed regardless of the memory's retrieval score.

Parsing failures and runtime cost. Memory evidence is parsed once when the memory is written, and its structured representation is cached with the trie index. At inference time, MEMTRIM only parses the current query once; retrieved memories reuse their cached evidence units, task signatures, and support sets. Trie matching therefore consists of key-value lookups over the evidence units in E_q and the retrieved memory identifiers.

We use constrained structured output and deterministic decoding for the parser. Each result is checked for valid fields, grounded source spans, and well-formed key-value types. If parsing fails, we retry once with the same interaction and the validation error. If a valid representation still cannot be produced, the raw memory remains in the base memory store but its stored outcome is not reused through MEMTRIM. This keeps parser failures from introducing an unchecked answer into the inference context.

B ADDITIONAL RESULTS

This appendix extends the main results in three directions. Appendices B.1 and B.2 repeat RQ2 and RQ3 of Section 2 on CounterLogic. Appendix B.3 repeats the mitigation comparison of Section 4 on the other three memory systems. Appendix B.4 walks through one case from each dataset to show how partial evidence overlap misleads ordinary memory use and how MEMTRIM corrects it.

B.1 RQ2 ON COUNTERLOGIC: WHERE FAILURES OCCUR

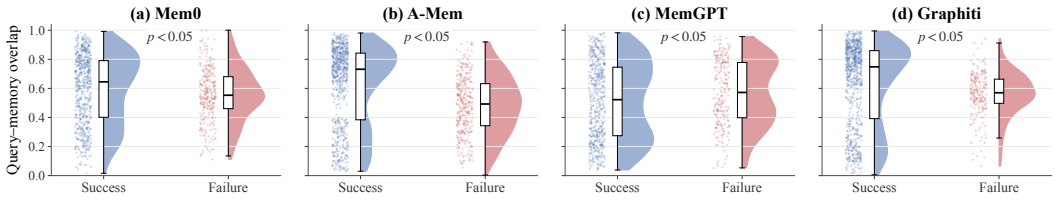

Figure 8: Query overlap in successful and failed cases on CounterLogic, for each memory system.

Figure 8 shows the same separation as Figure 2. Successful cases occur more often near the two ends of the overlap range, while failed cases concentrate at partial overlap.

B.2 RQ3 ON COUNTERLOGIC: CONTROLLED DUPLICATION

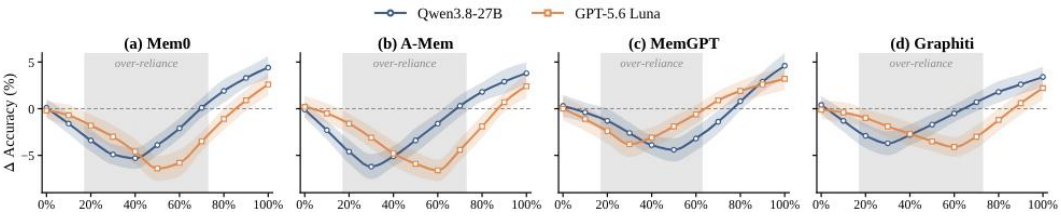

Figure 9: Accuracy change across levels of duplicated evidence on CounterLogic, for each memory system.

Figure 9 shows the same non-monotonic pattern as Figure 3. The accuracy loss is largest at intermediate duplication and disappears as duplication approaches full overlap. The drop is shallower than on RippleEdits, and the gain at full duplication is smaller.

B.3 MITIGATION ON OTHER MEMORY SYSTEMS

Table 2 fixes Mem0 as the memory backbone so that its rows differ only in the defense. Tables 3 and 4 replace Mem0 with A-Mem, MemGPT, and Graphiti, and keep the same baselines, datasets,

Table 3: Mitigation results (%) with Qwen3.8-27B on the other three memory systems. Each block uses the named system as the memory backbone, and its first two rows match Table 1. TA and TD-Dup follow Table 2, and ✓ →✗ and ✗ →✓ are counted on TD-Dup cases.

		RippleEdits				CounterLogic				CLADDER-Derived			
Memory	**Method**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**
A-Mem	No Mem.	68.0$_{\pm 0.4}$	71.3$_{\pm 0.8}$	–	–	85.4$_{\pm 1.3}$	86.1$_{\pm 0.9}$	–	–	60.1$_{\pm 0.5}$	61.2$_{\pm 0.9}$	–	–
	A-Mem	82.6$_{\pm 1.0}$	61.2$_{\pm 1.5}$	12.8	2.7	92.4$_{\pm 0.6}$	80.6$_{\pm 1.2}$	7.1	1.6	78.4$_{\pm 0.9}$	46.1$_{\pm 1.4}$	17.6	2.5
	Facts-only	71.0$_{\pm 0.5}$	72.0$_{\pm 0.4}$	2.4	3.1	86.8$_{\pm 0.8}$	86.8$_{\pm 1.4}$	1.3	2.0	63.6$_{\pm 1.1}$	61.3$_{\pm 1.9}$	3.1	3.2
	Top-1	73.5$_{\pm 2.0}$	61.7$_{\pm 0.6}$	12.0	2.4	88.2$_{\pm 1.8}$	80.5$_{\pm 1.7}$	7.1	1.5	66.7$_{\pm 1.6}$	47.7$_{\pm 0.8}$	15.8	2.3
	MMR	83.1$_{\pm 1.5}$	67.9$_{\pm 1.3}$	6.2	2.8	92.6$_{\pm 0.7}$	84.6$_{\pm 2.1}$	3.2	1.7	79.0$_{\pm 0.6}$	55.0$_{\pm 1.0}$	8.8	2.6
	Sim. filter	74.7$_{\pm 0.9}$	64.9$_{\pm 1.0}$	9.2	2.8	88.5$_{\pm 1.2}$	83.0$_{\pm 2.2}$	4.8	1.7	67.5$_{\pm 1.8}$	52.7$_{\pm 1.9}$	11.1	2.6
	RECOMP	82.0$_{\pm 0.7}$	61.4$_{\pm 1.1}$	12.5	2.6	92.4$_{\pm 1.3}$	80.5$_{\pm 1.9}$	7.2	1.6	76.8$_{\pm 1.7}$	47.3$_{\pm 1.4}$	16.3	2.4
	LLMLingua-2	77.3$_{\pm 1.2}$	59.8$_{\pm 0.7}$	14.0	2.5	90.4$_{\pm 1.8}$	78.9$_{\pm 0.6}$	8.7	1.5	72.7$_{\pm 1.1}$	42.9$_{\pm 0.5}$	20.7	2.4
	Context-faithful	83.6$_{\pm 2.0}$	72.3$_{\pm 2.3}$	1.5	2.5	92.8$_{\pm 1.3}$	86.8$_{\pm 2.1}$	0.8	1.5	79.6$_{\pm 0.9}$	61.5$_{\pm 1.2}$	2.0	2.3
	Answer-first	77.0$_{\pm 0.5}$	70.3$_{\pm 0.6}$	4.2	3.2	89.7$_{\pm 2.2}$	85.6$_{\pm 1.8}$	2.3	1.8	69.9$_{\pm 1.5}$	59.2$_{\pm 0.8}$	5.0	3.0
	Micro-Act	81.2$_{\pm 2.3}$	68.8$_{\pm 1.6}$	5.8	3.3	91.9$_{\pm 1.3}$	84.7$_{\pm 1.2}$	3.3	1.9	76.0$_{\pm 1.8}$	56.7$_{\pm 0.7}$	7.6	3.1
	MEMTRIM (ours)	**86.3$_{\pm 0.5}$**	**74.2$_{\pm 0.7}$**	**0.7**	**3.6**	**94.0$_{\pm 1.6}$**	**87.8$_{\pm 2.3}$**	**0.4**	**2.1**	**83.0$_{\pm 1.9}$**	**63.7$_{\pm 1.0}$**	**0.9**	**3.4**
MemGPT	No Mem.	68.0$_{\pm 0.4}$	71.3$_{\pm 0.8}$	–	–	85.4$_{\pm 1.3}$	86.1$_{\pm 0.9}$	–	–	60.1$_{\pm 0.5}$	61.2$_{\pm 0.9}$	–	–
	MemGPT	80.9$_{\pm 0.8}$	65.7$_{\pm 1.1}$	8.9	3.3	91.2$_{\pm 1.1}$	82.8$_{\pm 1.0}$	5.2	1.9	78.1$_{\pm 0.7}$	49.8$_{\pm 1.2}$	14.1	2.7
	Facts-only	70.5$_{\pm 2.2}$	73.4$_{\pm 1.4}$	1.6	3.7	86.6$_{\pm 2.1}$	87.4$_{\pm 1.2}$	1.0	2.3	63.8$_{\pm 1.0}$	62.0$_{\pm 0.7}$	2.6	3.4
	Top-1	73.2$_{\pm 1.2}$	65.5$_{\pm 1.5}$	8.8	3.0	87.6$_{\pm 2.0}$	83.0$_{\pm 2.3}$	4.9	1.8	67.3$_{\pm 1.3}$	49.7$_{\pm 0.8}$	14.0	2.5
	MMR	81.4$_{\pm 1.3}$	70.5$_{\pm 1.6}$	4.2	3.4	91.4$_{\pm 0.4}$	85.8$_{\pm 0.9}$	2.3	2.0	78.7$_{\pm 1.4}$	57.8$_{\pm 2.2}$	6.2	2.8
	Sim. filter	73.7$_{\pm 1.1}$	68.6$_{\pm 1.0}$	6.1	3.4	88.0$_{\pm 1.5}$	84.5$_{\pm 0.6}$	3.6	2.0	68.1$_{\pm 1.9}$	54.3$_{\pm 1.6}$	9.7	2.8
	RECOMP	80.5$_{\pm 1.0}$	65.7$_{\pm 0.4}$	8.8	3.2	91.0$_{\pm 0.9}$	82.9$_{\pm 1.1}$	5.1	1.9	77.2$_{\pm 1.6}$	50.2$_{\pm 1.4}$	13.6	2.6
	LLMLingua-2	76.9$_{\pm 1.3}$	63.9$_{\pm 2.2}$	10.5	3.1	89.3$_{\pm 1.2}$	82.0$_{\pm 2.1}$	5.9	1.8	72.9$_{\pm 1.1}$	46.7$_{\pm 1.5}$	17.1	2.6
	Context-faithful	82.0$_{\pm 1.8}$	73.4$_{\pm 2.0}$	1.0	3.1	91.7$_{\pm 1.3}$	87.3$_{\pm 0.6}$	0.6	1.8	79.4$_{\pm 1.9}$	62.0$_{\pm 0.4}$	1.7	2.5
	Answer-first	75.6$_{\pm 2.0}$	72.3$_{\pm 2.1}$	2.8	3.8	88.5$_{\pm 1.6}$	86.7$_{\pm 0.7}$	1.5	2.1	69.8$_{\pm 1.2}$	60.4$_{\pm 1.9}$	4.0	3.2
	Micro-Act	78.6$_{\pm 1.9}$	71.5$_{\pm 0.4}$	3.7	3.9	90.4$_{\pm 0.5}$	86.1$_{\pm 1.1}$	2.2	2.2	74.8$_{\pm 0.9}$	58.8$_{\pm 0.7}$	5.7	3.3
	MEMTRIM (ours)	**84.9$_{\pm 0.7}$**	**75.0$_{\pm 1.8}$**	**0.5**	**4.2**	**93.1$_{\pm 1.6}$**	**88.2$_{\pm 0.6}$**	**0.3**	**2.4**	**82.7$_{\pm 0.5}$**	**64.1$_{\pm 1.5}$**	**0.7**	**3.6**
Graphiti	No Mem.	68.0$_{\pm 0.4}$	71.3$_{\pm 0.8}$	–	–	85.4$_{\pm 1.3}$	86.1$_{\pm 0.9}$	–	–	60.1$_{\pm 0.5}$	61.2$_{\pm 0.9}$	–	–
	Graphiti	87.4$_{\pm 0.5}$	58.3$_{\pm 1.7}$	15.4	2.4	94.6$_{\pm 0.7}$	78.7$_{\pm 1.6}$	8.9	1.5	81.2$_{\pm 0.6}$	43.9$_{\pm 1.8}$	19.4	2.1
	Facts-only	71.9$_{\pm 2.3}$	71.3$_{\pm 1.8}$	2.8	2.8	87.2$_{\pm 1.1}$	86.4$_{\pm 1.5}$	1.6	1.9	64.1$_{\pm 1.4}$	60.6$_{\pm 0.6}$	3.4	2.8
	Top-1	74.8$_{\pm 1.1}$	60.0$_{\pm 1.9}$	13.4	2.1	89.0$_{\pm 1.5}$	78.8$_{\pm 2.2}$	8.7	1.4	67.9$_{\pm 2.1}$	45.3$_{\pm 2.0}$	17.8	1.9
	MMR	87.7$_{\pm 1.6}$	66.9$_{\pm 1.7}$	6.9	2.5	94.8$_{\pm 2.3}$	83.5$_{\pm 0.6}$	4.2	1.6	81.7$_{\pm 1.5}$	55.0$_{\pm 0.4}$	8.4	2.2
	Sim. filter	75.9$_{\pm 2.1}$	64.0$_{\pm 1.5}$	9.8	2.5	89.6$_{\pm 0.5}$	81.3$_{\pm 1.8}$	6.4	1.6	68.7$_{\pm 1.4}$	51.1$_{\pm 1.9}$	12.3	2.2
	RECOMP	85.6$_{\pm 1.8}$	59.4$_{\pm 1.4}$	14.2	2.3	93.6$_{\pm 1.6}$	79.5$_{\pm 1.3}$	8.1	1.5	78.9$_{\pm 1.9}$	45.6$_{\pm 0.8}$	17.6	2.0
	LLMLingua-2	80.1$_{\pm 0.4}$	57.1$_{\pm 0.5}$	16.4	2.2	91.4$_{\pm 1.0}$	77.6$_{\pm 1.8}$	9.9	1.4	74.2$_{\pm 0.8}$	41.0$_{\pm 1.2}$	22.2	2.0
	Context-faithful	88.2$_{\pm 0.7}$	71.6$_{\pm 1.5}$	1.9	2.2	94.9$_{\pm 2.2}$	86.5$_{\pm 0.5}$	1.0	1.4	82.3$_{\pm 1.0}$	60.8$_{\pm 1.6}$	2.3	1.9
	Answer-first	79.7$_{\pm 2.0}$	69.3$_{\pm 1.6}$	4.9	2.9	90.9$_{\pm 0.6}$	85.0$_{\pm 2.2}$	2.8	1.7	72.5$_{\pm 1.9}$	57.8$_{\pm 2.1}$	6.0	2.6
	Micro-Act	85.6$_{\pm 1.9}$	67.3$_{\pm 1.3}$	7.0	3.0	93.2$_{\pm 1.2}$	84.1$_{\pm 1.8}$	3.8	1.8	77.2$_{\pm 0.9}$	56.0$_{\pm 0.6}$	7.9	2.7
	MEMTRIM (ours)	**90.1$_{\pm 0.7}$**	**73.8$_{\pm 2.2}$**	**0.8**	**3.3**	**95.7$_{\pm 1.4}$**	**87.6$_{\pm 1.6}$**	**0.5**	**2.0**	**85.2$_{\pm 1.5}$**	**63.2$_{\pm 1.3}$**	**1.0**	**3.0**

and columns. The no-memory baseline does not depend on the memory system and is reported in Table 1.

The comparison with Table 2 holds for every backbone. The baselines that reduce how much memory reaches the model raise mismatch accuracy but lower match accuracy. The conflict-aware baselines improve mismatch accuracy while keeping match accuracy close to the backbone. MEMTRIM gives the highest mismatch accuracy and the lowest correct-to-wrong rate under every memory system, without lowering match accuracy.

B.4 CASE STUDIES

Each case shows the same failure from a different setting. The stored memory is correct for the query where it was written, while the current query changes information that determines the answer. We compare the context used by Mem0 with the context produced by MEMTRIM. Cases 1 and 3 remove the stored answer because its supporting evidence changes. Case 2 keeps the earlier result as labeled evidence because the task changes.

Table 4: Mitigation results (%) with GPT-5.6 Luna on the other three memory systems. Each block uses the named system as the memory backbone, and its first two rows match Table 1. TA and TD-Dup follow Table 2, and ✓ →✗ and ✗ →✓ are counted on TD-Dup cases.

		RippleEdits				CounterLogic				CLADDER-Derived			
Memory	**Method**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**	**TA**	**TD-Dup**	**✓ →✗**	**✗ →✓**
A-Mem	No Mem.	65.4$_{\pm 0.9}$	68.9$_{\pm 0.7}$	–	–	84.9$_{\pm 1.2}$	84.7$_{\pm 1.8}$	–	–	58.8$_{\pm 0.6}$	58.4$_{\pm 1.4}$	–	–
	A-Mem	83.7$_{\pm 1.0}$	60.8$_{\pm 1.1}$	10.9	2.8	92.2$_{\pm 0.9}$	80.1$_{\pm 1.0}$	6.4	1.8	78.9$_{\pm 1.1}$	46.2$_{\pm 1.6}$	14.8	2.6
	Facts-only	69.1$_{\pm 0.5}$	70.1$_{\pm 1.5}$	2.0	3.2	86.4$_{\pm 2.2}$	85.7$_{\pm 1.1}$	1.2	2.2	62.9$_{\pm 1.7}$	59.0$_{\pm 0.4}$	2.7	3.3
	Top-1	72.3$_{\pm 1.9}$	61.2$_{\pm 1.3}$	10.2	2.5	87.6$_{\pm 2.1}$	80.6$_{\pm 0.8}$	5.8	1.7	66.2$_{\pm 2.2}$	47.3$_{\pm 0.9}$	13.5	2.4
	MMR	84.1$_{\pm 2.0}$	67.0$_{\pm 2.2}$	4.8	2.9	92.4$_{\pm 0.9}$	83.7$_{\pm 1.9}$	2.9	1.9	79.4$_{\pm 2.1}$	54.7$_{\pm 0.4}$	6.4	2.7
	Sim. filter	73.1$_{\pm 1.9}$	64.7$_{\pm 0.6}$	7.1	2.9	88.2$_{\pm 1.6}$	82.1$_{\pm 1.7}$	4.5	1.9	68.2$_{\pm 0.8}$	50.4$_{\pm 0.5}$	10.7	2.7
	RECOMP	83.8$_{\pm 1.3}$	60.5$_{\pm 2.0}$	11.1	2.7	91.8$_{\pm 1.2}$	80.3$_{\pm 1.5}$	6.2	1.8	78.7$_{\pm 2.2}$	46.0$_{\pm 1.8}$	14.9	2.5
	LLMLingua-2	78.3$_{\pm 0.8}$	58.4$_{\pm 1.6}$	13.1	2.6	89.8$_{\pm 1.7}$	79.1$_{\pm 0.6}$	7.3	1.7	73.0$_{\pm 2.0}$	43.0$_{\pm 2.1}$	17.9	2.5
	Context-faithful	84.6$_{\pm 0.4}$	70.3$_{\pm 1.9}$	1.2	2.6	92.7$_{\pm 1.2}$	85.6$_{\pm 2.1}$	0.8	1.7	80.2$_{\pm 0.7}$	59.0$_{\pm 2.0}$	1.8	2.4
	Answer-first	76.1$_{\pm 2.2}$	68.8$_{\pm 1.6}$	3.4	3.3	89.1$_{\pm 1.4}$	84.8$_{\pm 0.9}$	1.9	2.0	70.8$_{\pm 1.3}$	56.9$_{\pm 1.0}$	4.6	3.1
	Micro-Act	81.6$_{\pm 0.9}$	67.5$_{\pm 2.2}$	4.8	3.4	90.9$_{\pm 2.1}$	84.2$_{\pm 1.4}$	2.6	2.1	76.7$_{\pm 1.5}$	55.0$_{\pm 0.7}$	6.6	3.2
	MEMTRIM (ours)	**87.1**$_{\pm 1.0}$	**72.0**$_{\pm 1.4}$	**0.6**	**3.7**	**93.8**$_{\pm 1.9}$	**86.7**$_{\pm 1.1}$	**0.3**	**2.3**	**83.3**$_{\pm 1.8}$	**61.1**$_{\pm 1.2}$	**0.8**	**3.5**
MemGPT	No Mem.	65.4$_{\pm 0.9}$	68.9$_{\pm 0.7}$	–	–	84.9$_{\pm 1.2}$	84.7$_{\pm 1.8}$	–	–	58.8$_{\pm 0.6}$	58.4$_{\pm 1.4}$	–	–
	MemGPT	82.4$_{\pm 0.5}$	65.5$_{\pm 0.9}$	6.8	3.4	91.6$_{\pm 0.6}$	82.1$_{\pm 0.8}$	4.6	2.0	79.2$_{\pm 1.0}$	48.6$_{\pm 1.1}$	12.6	2.8
	Facts-only	68.8$_{\pm 2.3}$	71.5$_{\pm 2.1}$	1.2	3.8	86.3$_{\pm 1.2}$	86.2$_{\pm 0.9}$	0.9	2.4	62.9$_{\pm 1.1}$	59.6$_{\pm 1.5}$	2.3	3.5
	Top-1	71.3$_{\pm 2.0}$	66.1$_{\pm 1.0}$	5.9	3.1	87.5$_{\pm 0.9}$	82.2$_{\pm 2.1}$	4.4	1.9	66.5$_{\pm 0.4}$	49.2$_{\pm 0.7}$	11.8	2.6
	MMR	82.8$_{\pm 1.4}$	69.4$_{\pm 0.8}$	3.0	3.5	91.8$_{\pm 0.7}$	84.5$_{\pm 2.2}$	2.3	2.1	79.8$_{\pm 1.8}$	55.5$_{\pm 1.0}$	5.8	2.9
	Sim. filter	72.8$_{\pm 1.6}$	67.8$_{\pm 1.7}$	4.6	3.5	88.1$_{\pm 0.4}$	83.4$_{\pm 0.7}$	3.4	2.1	67.2$_{\pm 0.5}$	53.2$_{\pm 1.9}$	8.1	2.9
	RECOMP	80.3$_{\pm 0.6}$	66.1$_{\pm 0.7}$	6.1	3.3	91.5$_{\pm 1.7}$	82.1$_{\pm 1.5}$	4.6	2.0	79.0$_{\pm 1.8}$	48.4$_{\pm 0.9}$	12.7	2.7
	LLMLingua-2	75.9$_{\pm 1.3}$	64.9$_{\pm 1.0}$	7.2	3.2	89.3$_{\pm 0.7}$	81.5$_{\pm 1.1}$	5.1	1.9	72.1$_{\pm 1.9}$	47.1$_{\pm 2.0}$	14.0	2.7
	Context-faithful	83.4$_{\pm 1.7}$	71.3$_{\pm 0.6}$	0.8	3.2	92.1$_{\pm 0.8}$	86.1$_{\pm 2.2}$	0.5	1.9	80.4$_{\pm 1.2}$	59.5$_{\pm 2.3}$	1.5	2.6
	Answer-first	75.9$_{\pm 1.0}$	70.6$_{\pm 1.3}$	2.2	3.9	88.7$_{\pm 2.3}$	85.5$_{\pm 2.0}$	1.4	2.2	69.7$_{\pm 1.4}$	58.1$_{\pm 2.2}$	3.6	3.3
	Micro-Act	81.1$_{\pm 1.7}$	69.8$_{\pm 2.1}$	3.1	4.0	90.7$_{\pm 1.0}$	85.0$_{\pm 1.1}$	2.0	2.3	77.7$_{\pm 0.5}$	56.0$_{\pm 1.9}$	5.8	3.4
	MEMTRIM (ours)	**86.1**$_{\pm 0.5}$	**72.8**$_{\pm 0.8}$	**0.4**	**4.3**	**93.4**$_{\pm 1.4}$	**87.0**$_{\pm 1.3}$	**0.2**	**2.5**	**83.6**$_{\pm 1.1}$	**61.4**$_{\pm 0.7}$	**0.7**	**3.7**
Graphiti	No Mem.	65.4$_{\pm 0.9}$	68.9$_{\pm 0.7}$	–	–	84.9$_{\pm 1.2}$	84.7$_{\pm 1.8}$	–	–	58.8$_{\pm 0.6}$	58.4$_{\pm 1.4}$	–	–
	Graphiti	88.0$_{\pm 0.4}$	57.8$_{\pm 1.5}$	13.6	2.5	94.3$_{\pm 0.5}$	77.9$_{\pm 1.4}$	8.3	1.5	82.1$_{\pm 0.6}$	43.5$_{\pm 1.4}$	17.1	2.2
	Facts-only	69.8$_{\pm 2.0}$	69.4$_{\pm 1.6}$	2.4	2.9	86.8$_{\pm 1.9}$	85.1$_{\pm 1.8}$	1.5	1.9	63.3$_{\pm 1.4}$	58.3$_{\pm 1.3}$	3.0	2.9
	Top-1	73.7$_{\pm 1.7}$	58.8$_{\pm 2.0}$	12.3	2.2	88.7$_{\pm 1.5}$	77.8$_{\pm 1.3}$	8.3	1.4	67.3$_{\pm 2.2}$	45.0$_{\pm 1.8}$	15.4	2.0
	MMR	88.3$_{\pm 2.0}$	65.5$_{\pm 1.9}$	6.0	2.6	94.5$_{\pm 1.7}$	82.2$_{\pm 0.8}$	4.1	1.6	82.6$_{\pm 1.3}$	52.2$_{\pm 0.6}$	8.5	2.3
	Sim. filter	74.8$_{\pm 1.6}$	62.8$_{\pm 2.3}$	8.7	2.6	88.9$_{\pm 0.7}$	80.8$_{\pm 2.0}$	5.5	1.6	68.6$_{\pm 0.6}$	49.5$_{\pm 2.2}$	11.2	2.3
	RECOMP	87.3$_{\pm 2.0}$	57.9$_{\pm 2.2}$	13.4	2.4	93.7$_{\pm 1.2}$	78.3$_{\pm 1.5}$	7.9	1.5	79.0$_{\pm 1.4}$	45.4$_{\pm 0.7}$	15.1	2.1
	LLMLingua-2	79.6$_{\pm 0.8}$	56.5$_{\pm 1.9}$	14.7	2.3	91.6$_{\pm 0.6}$	76.0$_{\pm 1.0}$	10.1	1.4	73.8$_{\pm 1.2}$	41.7$_{\pm 1.6}$	18.8	2.1
	Context-faithful	88.7$_{\pm 2.1}$	69.7$_{\pm 1.8}$	1.5	2.3	94.6$_{\pm 1.0}$	85.1$_{\pm 0.9}$	1.0	1.4	83.1$_{\pm 0.5}$	58.5$_{\pm 1.1}$	1.9	2.0
	Answer-first	77.3$_{\pm 1.0}$	68.1$_{\pm 1.9}$	3.8	3.0	90.6$_{\pm 0.5}$	83.8$_{\pm 0.7}$	2.6	1.7	71.6$_{\pm 0.9}$	56.1$_{\pm 2.3}$	5.0	2.7
	Micro-Act	84.9$_{\pm 2.2}$	66.1$_{\pm 1.3}$	5.9	3.1	93.3$_{\pm 0.6}$	82.8$_{\pm 1.9}$	3.7	1.8	78.7$_{\pm 1.6}$	53.9$_{\pm 0.9}$	7.3	2.8
	MEMTRIM (ours)	**90.5**$_{\pm 0.7}$	**71.6**$_{\pm 0.5}$	**0.7**	**3.4**	**95.5**$_{\pm 1.8}$	**86.3**$_{\pm 0.6}$	**0.4**	**2.0**	**85.9**$_{\pm 1.7}$	**60.6**$_{\pm 1.0}$	**0.9**	**3.1**

Case 1 RippleEdits · Outdated knowledge

Stored memory

Query: Lionel Messi plays for Paris Saint-Germain. Which country is his club based in?
Answer: France.

Current query

Lionel Messi now plays for Inter Miami. Which country is his club based in?

Mem0

Mem0 passes the retrieved memory in full, including the stored answer "France."

Prediction: France ✗

MEMTRIM

Shared	(`person`, Lionel Messi); task $\tau_m = \tau_q =$ `club_country`.
Changed	(`club`, Paris Saint-Germain) → (`club`, Inter Miami).
Memory-only	(`position`, forward).
Decision	The stored answer depends on the club. Since this evidence changed, MEMTRIM removes the stored answer.

Context passed to the model: (`position`, forward).

Prediction: United States ✓

Why Mem0 fails

The person and task remain the same, so the memory looks relevant. The club changes, and this change determines the correct answer. Mem0 still exposes the old answer, which pulls the prediction toward France.

Case 2 CounterLogic · Query change

Stored memory
Premises: All metals are liquid at room temperature. Iron is a metal.
Conclusion: Iron is liquid at room temperature.
Query: Is the argument logically valid?
Answer: Yes.

Current query
The premises and conclusion remain unchanged.
Query: Is the conclusion true in the real world?

Mem0
Mem0 passes the retrieved memory in full, including the stored answer "Yes."

Prediction: Yes ✗

MEMTRIM

Shared	Both premises and the conclusion.
Changed	The task changes from `validity` to `real_world_truth`.
Memory-only	None.
Decision	The evidence itself does not conflict with the current query. MEMTRIM keeps the earlier result as a labeled validity statement rather than presenting it as the current answer.

Context passed to the model: (`validity`, yes).

Prediction: No ✓

Why Mem0 fails
The premises and conclusion are identical across the two queries, so the retrieved memory has very high overlap with the current input. The question changes from logical validity to real-world truth. Mem0 presents the earlier "Yes" without marking which question it answers, which encourages the model to reuse it.

Case 3 CLADDER-anti · Premise change

Stored memory
Setup: Eating citrus (X) directly affects scurvy (Y). Genotype (Z) affects both X and Y.
$P(Y{=}1 \mid X{=}1) = 0.72$, $P(Y{=}1 \mid X{=}0) = 0.41$.
Query: Does eating citrus increase the chance of scurvy?
Answer: Yes.

Current query
The variables and graph remain the same. The query now gives the interventional probabilities

$$P(Y{=}1 \mid do(X{=}1)) = 0.38, \qquad P(Y{=}1 \mid do(X{=}0)) = 0.41.$$

Query: Does eating citrus increase the chance of scurvy?

Mem0
Mem0 passes the retrieved memory in full, including the stored answer "Yes."

Prediction: Yes ✗

MEMTRIM

Shared	The variable definitions, causal graph, and task $\tau_m = \tau_q =$ `causal_effect`.
Changed	The answer-relevant probabilities change from observational values to interventional values.
Memory-only	None.
Decision	The stored answer depends on the earlier probabilities. Since these values change, MEMTRIM removes the stored answer.

Context passed to the model: No additional memory content is needed beyond the current query.

Prediction: No ✓

Why Mem0 fails
Most of the causal setup is repeated, so the retrieved memory looks like the same problem. The probabilities that determine the answer change. Mem0 still exposes the earlier "Yes," which pulls the model toward the stored conclusion.

Across all three cases, most of the retrieved memory remains related to the current query, while the information that determines the answer changes. This combination makes the earlier answer easy to reuse. MEMTRIM separates shared, changed, and memory-only information before the memory reaches the model. It then uses the evidence supporting the stored answer to decide whether that answer should be kept, relabeled, or removed.

C CONSTRUCTION OF TASK-ALIGNED AND TASK-DIVERGENT MEMORY PAIRS

The source benchmarks were not designed for agentic memory evaluation and do not directly provide a stored interaction paired with a later query. We derive such pairs from the released benchmark annotations. A *task-aligned* (TA) pair uses a stored answer that remains valid for the current query. A *task-divergent with duplicated evidence* (TD-Dup) pair preserves substantial information from the stored interaction but changes the task or answer-relevant information, so the stored answer no longer applies.

No LLM is used to generate examples, assign TA/TD-Dup labels, or determine gold answers. Labels come from the original benchmark annotations or fixed rules described below. Sentence encoders are used only to identify related candidate examples when an explicit pair is not provided by the benchmark.

C.1 RIPPLEEDITS

RippleEdits (Cohen et al., 2024) represents factual changes as subject-relation-object edits and provides both the original and updated values for modification examples. We use the RANDOM and POPULAR subsets and exclude RECENT, which contains fact insertions without the corresponding old value.

Dataset	Fixed across pair	Changed information	TA	TD-Dup
RippleEdits	Entity and relation	Entity value	Updated value	Original value
CounterLogic	Premises and conclusion	Question being answered	Same gold answer	Different gold answer
CLADDER-derived	Story, graph, causal task	Causal values	Same gold answer	Different gold answer

Table 5: Construction of TA and TD-Dup pairs. Both conditions preserve the same underlying scenario and substantial shared information, but differ in whether the stored answer remains valid for the current query.

For each edit, we keep the entity, relation, and current query fixed. The TA memory stores the updated value, while the TD-Dup memory stores the original value. Thus, both memories refer to the same fact but differ in the value that determines the answer. We use Subject Aliasing queries because they preserve the underlying subject-relation pair while varying only how the subject is mentioned.

We remove examples when either target is missing or when the original and updated targets become indistinguishable after alias normalization. From the 2,000 RANDOM and POPULAR edits, we sample at most 500 source queries with random seed 42. Each retained query contributes one TA and one TD-Dup condition.

C.2 COUNTERLOGIC

CounterLogic (Joishy et al., 2025) provides each set of premises and conclusion with two judgments: logical validity and agreement with real-world knowledge. This lets us change the question while keeping the underlying evidence fixed.

For each instance, the stored interaction asks whether the conclusion is logically valid, while the current interaction asks whether it agrees with real-world knowledge. We use fixed templates for both questions and copy the premises and conclusion directly from the benchmark. An instance is TA when the validity and knowledge-alignment labels agree, and TD-Dup when they differ. The pair type therefore comes entirely from the two benchmark labels.

To form plausible retrieval groups, we encode the premises and conclusion with `sentence-transformers/all-mpnet-base-v2`, excluding the task instruction, gold label, and any answer-revealing fields. We use agglomerative clustering with average linkage and cosine distance, a distance threshold of 0.20, and a minimum cluster size of five. Clustering only limits which memories can be retrieved; it does not determine whether an example is TA or TD-Dup.

We use all 1,800 CounterLogic instances and sample at most 500 TA and 500 TD-Dup examples with random seed 42.

C.3 CLADDER-DERIVED PREMISE-CHANGE PAIRS

CLADDER (Jin et al., 2023) provides natural-language causal questions, structured metadata, and gold answers. We derive premise-change pairs from the official balanced release while keeping the original question text and gold labels.

We first group examples that have the same `story_id`, `graph_id`, `query_type`, and `rung`. This keeps the causal scenario and question type fixed. Within each group, we replace all numerical values with a common `<NUM>` token and encode the remaining text with `sentence-transformers/all-mpnet-base-v2`. Candidate pairs must have cosine similarity of at least 0.85.

We then verify that the pair also overlaps at the evidence level using the structured causal quantities provided by CLADDER. Each numerical statement is converted to a key-value entry. We retain pairs whose evidence keys have Jaccard similarity of at least 0.80, contain at least one unchanged value and at least one changed value, and have at least one changed value that appears in the benchmark-provided reasoning for the queried causal quantity. These checks ensure that the two examples share most of the same causal setup while differing in information that can affect the answer.

The original CLADDER answers determine the pair type: a pair is TA when the two gold answers agree and TD-Dup when they differ. The sentence encoder and evidence-overlap checks are used only to decide whether two examples are similar enough to form a pair.

For each source interaction, we select one TA and one TD-Dup candidate with similar semantic similarity and evidence overlap. When several candidates are available, we choose the pair with the smallest combined difference in these two quantities and break ties by example identifier. Source interactions without suitable candidates for both conditions are removed. From the 10,112 questions in the official release, we sample at most 500 TA and 500 TD-Dup pairs with random seed 42.

C.4 Construction Checks

We apply the same checks across all three datasets. Every stored interaction must be correct in the setting where it was written, and every current query must have a gold answer supplied by the benchmark or its formal reasoning procedure. Model predictions are never used as labels.

We remove duplicate memory-query pairs and keep examples derived from the same source unit in the same data split. We also compare query-memory similarity, evidence overlap when available, and input length between TA and TD-Dup after construction. If the two groups differ substantially on these properties, we apply additional matching before evaluation.

Task alignment only describes whether the stored answer remains valid for the current query. It does not require memory to be necessary for solving the query.