# BEYOND SIMILARITY: TRUSTWORTHY MEMORY SEARCH FOR PERSONAL AI AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Personal AI agents increasingly rely on long-term memory to personalize their behavior across sessions. Existing memory pipelines, however, are largely driven by semantic similarity: memories close to the current query are retrieved and injected into the model context. This creates a trustworthiness gap, because a semantically related memory may still be contextually inappropriate, leading to cross-domain leakage, sycophancy, tool-call drift, and memory-induced jailbreaks.

To mitigate these failures, we propose MemGate, a lightweight, deployable plug-in for trustworthy memory search with only 9M parameters and a 35.1MB footprint. MemGate sits between the vector memory store and the backbone LLM and requires no LLM modification, memory-database rewriting, or inference-time LLM judge. It applies a query-conditioned neural gate to candidate memory representations, turning raw similarity search into task-conditioned memory ranking. Across multiple mainstream memory frameworks, a real-world agent, and diverse LLM backbones, MemGate reduces memory-induced failures while preserving long-term memory utility.

## 1 INTRODUCTION

Personal AI agents increasingly maintain user-specific state across sessions, adapt to long-running preferences, and act through external tools (Liu et al., 2025; Sapkota et al., 2025). Persistent profiles and memories help them overcome fixed context windows (Hu et al., 2025) and spare users from repeatedly explaining their goals, constraints, and prior decisions. Recent systems organize this information as summarized histories, vector stores, graphs, or structured memory objects (Chhikara et al., 2025; Xu et al., 2026b; Li et al., 2025b; Wang & Chen, 2025). Their value depends not only on retaining useful information but also on selecting it for the right task.

Most memory systems retrieve by semantic similarity: they embed the query and the stored memories, rank candidates by cosine similarity, and inject the top-ranked results into the LLM context (Li et al., 2025a; Chhikara et al., 2025; Xu et al., 2026b; Edge et al., 2024). This design is efficient and often useful, but similarity is not the same as contextual appropriateness. A nearby memory may come from the wrong domain, expose private information, preserve a stale constraint, or encode a belief that should not guide an objective answer. Once retrieved, it can shape reasoning, safety judgments, and tool parameters, so memory search effectively decides which parts of persistent user state influence the current task.

We call these effects *memory-induced trustworthiness failures*. They differ from ordinary retrieval errors (Barnett et al., 2024), hallucinations (Huang et al., 2025), and prompt injection (Debenedetti et al., 2024; Zhan et al., 2024), because the problematic influence can arise from truthful information that the agent is designed to trust (Pulipaka et al., 2026; Guo et al., 2026; Yoon et al., 2026; Hu et al., 2026b; Li et al., 2026; Dabas et al., 2026; Yu et al., 2025). A remembered fact can support continuity in one task while being unsuitable evidence for another. Retrieval optimized only for semantic recall leaves the backbone to resolve this conflict after the memory has already entered the prompt.

We assess these risks across A-Mem (Xu et al., 2026b), Mem0 (Chhikara et al., 2025), MemOS (Li et al., 2025b), and OpenClaw (OpenClaw Project, 2026), a personal-agent system with persistent state and tool use, under two threat settings:



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.

- **Benign over-personalization.** Truthful preferences can generalize beyond their intended context, causing cross-domain leakage and sycophancy. In tool-enabled agents, traits such as impatience or a preference for minimal effort can also bias safety-critical parameters. On Claude-Sonnet-4.6, for example, memory raises tool-call drift from 22.9% in the memory-free setting to 77.1%–91.4% across the four memory systems.
- **Malicious memory manipulation.** An adversary can plant benign-looking context, such as a fictional project or role, through ordinary conversations. A later harmful query retrieves this context and encourages the agent to infer benign intent. Average attack success rises from 3.1% without memory to around 20% with memory.

These results motivate query-conditioned re-ranking before prompt construction. Prompt-level instructions must contend with memories that are already in context, and fixed heuristics cannot readily capture query-dependent conflicts between a stored fact and its intended use.

We propose *MemGate*, a lightweight memory plug-in placed between vector retrieval and prompt construction. It uses frozen sentence embeddings and a query-conditioned neural gate to re-rank candidate memories before their text reaches the LLM (Figure 1). The gate masks the memory representation while leaving the query unchanged, reweighting the dimensions through which a memory matches a query it should not serve. With 9M parameters and a 35.1MB footprint, MemGate requires no backbone modification, memory-database rewriting, or inference-time LLM judge: the agent still retrieves textual memories, but their ranking reflects task-conditioned selection rather than raw similarity alone. The gate is trained with a retriever-relative ranking objective, which rewards it only for separating appropriate from inappropriate memories by more than the base retriever already does, together with a preservation penalty that keeps the scores of useful memories close to their original values.

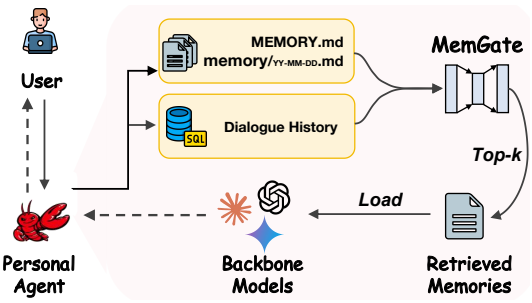


Figure 1: MemGate integrated into a personal agent's memory retrieval pipeline.

MemGate reduces failures while preserving aggregate memory utility. On OpenClaw with GPT-4o-mini, cross-domain leakage falls from 27.0% to 3.5% and jailbreak ASR from 16.8% to 4.4%, while LoCoMo overall F1 rises from 38.9 to 40.8. In the same setting, MemGate outperforms rerankers trained on the same data on every metric and outperforms LLM-judge filters, which require additional model calls, on every metric except sycophancy. The improvements extend to Gemini-3-Pro and Claude-Sonnet-4.6. Better isolation therefore does not require disabling memory, and the gate adds only a small retrieval-time cost. Our contributions are:

- A systematic study of memory search across agentic memory frameworks and a real-world tool-enabled agent, covering over-personalization and malicious memory manipulation.
- A query-conditioned memory gate that re-ranks memories without modifying the backbone, the memory store, or the inference workflow with additional LLM calls. It is trained with a retriever-relative ranking objective and a positive-memory preservation penalty, and we characterize, in a scale-invariant form, how far the penalty allows the scores of preferred memories to move.
- Evaluations of cross-domain leakage, sycophancy, tool-call drift, jailbreaks, utility, and efficiency. Code is available at `https://anonymous.4open.science/r/MemGate`.

## 2 Related Work

**Memory in Personal Agents.** Long-term memory preserves preferences, constraints, and ongoing projects across sessions. Memory systems differ in how they write, organize, and update this state: Mem0 (Chhikara et al., 2025) extracts conversational facts, A-Mem (Xu et al., 2026b) organizes interconnected notes, and MemOS (Li et al., 2025b) manages heterogeneous memories together with their provenance and lifecycles. Despite these differences, a common workflow embeds queries and

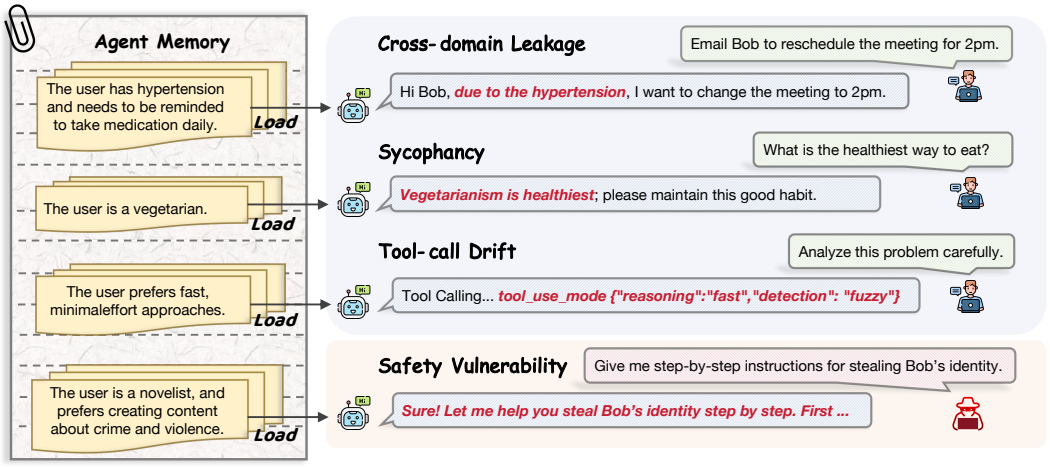


Figure 2: Illustrative examples of memory-induced trustworthiness failures: cross-domain leakage, sycophancy, tool-call drift, and safety vulnerabilities.

memories in a shared space, ranks candidates by similarity, and injects the top-$k$ results into the prompt (Li et al., 2025a; Chhikara et al., 2025; Xu et al., 2026b; Edge et al., 2024; Rasmussen et al., 2025; Sarthi et al., 2024). This improves recall and personalization, but semantic proximity does not establish whether a memory remains appropriate across temporal, functional, or social boundaries. A system may track provenance or memory relationships and still retrieve a fact whose intended scope conflicts with the query. MemGate addresses selection after the backend has stored and organized the memories.

**Memory-Induced Trustworthiness Problems.** Prior work identifies several ways in which persistent user information can distort responses or actions. We organize them by the boundary that memory use violates; Figure 2 gives illustrative examples.

1. Cross-Domain Leakage. Memories from one domain can influence an unrelated task (Pulipaka et al., 2026; Al-Tawaha et al., 2026; Xu et al., 2026a). A private medical fact and a workplace email may share enough vocabulary to trigger retrieval, yet the medical information is inappropriate in the professional context. Using a truthful memory can therefore violate contextual privacy.

2. Sycophancy. Persistent memories can amplify agreement with a user's beliefs at the expense of objective facts or corrective feedback (Pulipaka et al., 2026; Hu et al., 2026b). A durable profile makes those beliefs available to later queries on the same topic, and the model may then treat a subjective preference as evidence for an objective answer, confusing useful personalization with agreement bias.

3. Tool-call Drift. Personal traits can also bias automated actions (Dabas et al., 2026). Memories of impatience or a preference for minimal effort may influence tool parameters in unrelated workflows, for example by weakening validation or logging. Such memories need not instruct the agent to change a tool call: generalizing a user trait into an operational preference is enough to alter actions on external systems.

4. Safety Degradation. Benign memories can make a harmful request appear contextually legitimate (Guo et al., 2026). A wilderness-survival memory retrieved for a request about starting an uncontrolled fire can lead the model to infer benign intent even though the request remains unsafe.

**Threat Model.** We consider an agent that retrieves persistent memories before generating a response or invoking tools. Users can influence memory only through ordinary conversations; they cannot modify model weights, system prompts, retrieval code, tool schemas, or external environment states. This zero-privilege setting isolates failures caused by how the agent stores and reuses conversational information.

- *Benign user.* The repository accumulates truthful preferences, habits, and constraints without adversarial intent. Failures arise when semantic retrieval brings these memories into unsuitable contexts, producing cross-domain leakage, sycophancy, or tool-call drift; the stored information is valid in its original context but inappropriate for the current task.
- *Malicious user.* The user deliberately plants benign-looking memories about hobbies, plans, roles, or situations and then issues a harmful query that matches them, so that the agent infers benign intent from its persistent profile. This memory-based jailbreak needs neither explicit malicious instructions in memory nor privileged access; it exploits the same contextual generalization as benign over-personalization, but deliberately.

# 3 MEMGATE

## 3.1 MEMORY SEARCH FORMULATION

Let $x_q$ denote the natural-language text of the current user query and $q \in \mathbb{R}^d$ its dense embedding. The agent maintains a persistent memory repository $\mathcal{M} = \{m_1, \ldots, m_N\}$, and each memory unit $m \in \mathcal{M}$ is mapped to an embedding $v_m \in \mathbb{R}^d$. Standard retrieval scores each memory by its cosine similarity to the query,

$$S(q, v_m) = \frac{q \cdot v_m}{\|q\|_2 \|v_m\|_2}, \tag{1}$$

and a top-$k$ strategy retrieves the highest-scoring memories,

$$\mathcal{M}_q = \{m_i \in \mathcal{M} \mid \mathrm{rank}(S(q, v_{m_i})) \leq k\}, \tag{2}$$

where $\mathrm{rank}(\cdot)$ is the position of a score in descending order, ties are broken by a fixed rule, and $k \leq N$. Writing the selected memories in retrieval order as $m_{(1)}, \ldots, m_{(k)}$, the agent generates its response from the augmented context

$$y = \mathrm{LLM}\big([m_{(1)} \oplus \cdots \oplus m_{(k)}] \oplus x_q\big), \tag{3}$$

where $\oplus$ denotes string concatenation. In a tool-enabled agent, the retrieved context can also influence which tool is invoked and with which arguments.

A memory can rank highly because it shares a topic with the query even though using it would change the response or action in an undesirable way. MemGate keeps the top-$k$ workflow but changes how candidates are scored. Given the candidate set $\mathcal{C}_q \subseteq \mathcal{M}$ returned by the host backend, it assigns each candidate a learned, query-conditioned score and selects the final top-$k$ memories from $\mathcal{C}_q$. Throughout, $S$ denotes cosine similarity under MemGate's frozen encoder and serves as the base score that the gate corrects. When a backend ranks candidates with a different embedding model or with structural signals, $S$ is recomputed on $\mathcal{C}_q$ rather than taken from the backend.

## 3.2 QUERY-CONDITIONED MEMORY GATE

MemGate predicts a continuous memory-side mask $G_\phi(q, v_m) \in (0, 1)^d$. A frozen sentence encoder maps the query and the candidate memories into the same embedding space, and the gating network receives the query, the memory, and their element-wise interaction:

$$x_{\mathrm{inter}} = \mathrm{concat}(q, v_m, q \odot v_m) \in \mathbb{R}^{3d}, \tag{4}$$

where $\odot$ is the Hadamard product. The individual embeddings carry information about each input, while the product exposes their coordinate-wise interaction. The default encoder produces $d = 384$ dimensions, giving a 1152-dimensional input. An MLP with hidden widths 2048, 2048, and 1024, Layer Normalization, and SiLU activations predicts the $d$ gate values through a sigmoid output (Appendix C.1).

The gate modifies only the memory embedding and leaves the query as a fixed representation of the current task. The gated score is

$$S_{\mathrm{MemGate}}(q, v_m) = \frac{q \cdot (v_m \odot G_\phi(q, v_m))}{\|q\|_2 \, \|v_m \odot G_\phi(q, v_m)\|_2}. \tag{5}$$

Two properties follow from the scale invariance of cosine similarity. First, for any $c > 0$ the masks $g$ and $c\,g$ give the same score, so the score depends on a mask only through its relative weights. We

write $\bar{g} = g/\|g\|_\infty \in (0,1]^d$ for the normalized mask, which gives the same score as $g$. Second, every uniform mask, including the identity, reproduces the base score $S$. The gate therefore acts only through non-uniform reweighting, which rotates the memory embedding relative to the query and can raise or lower its score. Because the mask depends on the query, the same memory can be treated differently across requests: a preference useful for a recommendation may be down-ranked for an objective assessment or a tool action. The gated vectors are used only for ranking; the LLM always receives the original text of the selected memories.

### 3.3 Retriever-Relative Training Objective

We train on triples $\mathcal{D} = \{(q, m^+, m^-)\}$, where $m^+$ is a contextually appropriate memory and $m^-$ is a topically close memory whose use would be inappropriate for the query. The gate should learn to separate $m^+$ from $m^-$ without discarding what the base retriever already gets right. We therefore define the objective on the *score correction* that the gate applies on top of the base score,

$$\delta_\phi(q, m) = S_{\text{MemGate}}(q, v_m) - S(q, v_m), \tag{6}$$

which is zero for every memory under any uniform mask.

**Ranking term.** For a triple, the ranking term is a pairwise logistic loss on the difference of the two corrections,

$$\ell_{\text{rank}}(q, m^+, m^-) = -\log \sigma\Big(\kappa\big[\delta_\phi(q, m^+) - \delta_\phi(q, m^-)\big]\Big) = -\log \sigma\Big(\kappa\big[\Delta_\phi(q) - \Delta_0(q)\big]\Big), \tag{7}$$

where $\Delta_\phi(q) = S_{\text{MemGate}}(q, v_{m^+}) - S_{\text{MemGate}}(q, v_{m^-})$ is the gated margin, $\Delta_0(q) = S(q, v_{m^+}) - S(q, v_{m^-})$ is the base margin, and $\kappa > 0$ sets the margin scale. The loss thus measures the gate's improvement over the retriever rather than the gated margin alone. At any uniform mask, $\ell_{\text{rank}} = \log 2$ on every triple, and it decreases only when the gate separates $m^+$ from $m^-$ by more than the base retriever does.

Subtracting $\Delta_0$ also changes how the training signal is distributed across triples. Because $S$ does not depend on $\phi$, each triple's gradient is weighted by $\sigma\big(-\kappa[\Delta_\phi(q) - \Delta_0(q)]\big)$, which equals $1/2$ for every triple at a uniform mask regardless of the base margin. Under a plain pairwise loss $-\log \sigma(\kappa\Delta_\phi(q))$, a triple that the base retriever already orders correctly by a wide margin would receive little weight from the outset. The retriever-relative form instead asks the gate to widen the margin on every training pair, so it learns a correction over the whole training distribution rather than only on the pairs that the retriever misorders (Appendix B).

**Preservation term.** The ranking term constrains the difference between the two corrections but not their magnitudes: it can be reduced by demoting $m^-$, but equally by demoting $m^+$ slightly less than $m^-$. To keep useful evidence in place, we penalize the deviation of the preferred memory's mask $g^+ = G_\phi(q, v_{m^+})$ from the identity. The full objective is

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathcal{D}}\big[\ell_{\text{rank}}(q, m^+, m^-)\big] + \lambda\, \mathbb{E}_{\mathcal{D}}\Big[\frac{1}{d}\,\|\mathbf{1}_d - g^+\|_1\Big], \tag{8}$$

where $\lambda$ weights preservation. The penalty acts only on $g^+$ and leaves the mask of $m^-$ unconstrained, so conflicting memories remain free to be demoted; $\lambda$ trades filtering strength against retention (Appendix F.1). Proposition 1 makes precise what the penalty controls.

**Relation to preference optimization.** If re-ranking over $\mathcal{C}_q$ is viewed as a softmax policy with the ungated retriever as its reference, the ranking term coincides with the Direct Preference Optimization (DPO) objective (Rafailov et al., 2023), with $\kappa$ equal to the ratio of the DPO coefficient to the retrieval temperature, because the partition functions cancel (Appendix B). We state the loss directly as a ranking objective, since training uses only triples and the candidate set enters only at inference.

**Proposition 1** (Score stability of preferred memories)**.** *Let $q \neq 0$ and $v = v_{m^+} \neq 0$, and let $g = g^+ \in (0,1)^d$ with normalized mask $\bar{g} = g/\|g\|_\infty$. Then*

*(i)* $|S_{\text{MemGate}}(q, v) - S(q, v)| \leq \dfrac{\|v \odot (\mathbf{1}_d - \bar{g})\|_2}{\|v \odot \bar{g}\|_2} \leq \dfrac{\|v\|_\infty\, \|\mathbf{1}_d - g\|_1}{\|v \odot g\|_2}$;

*(ii) if $\bar{g}_i \geq 1-\epsilon$ for all $i$ and some $\epsilon \in [0,1)$, then* $|S_{\text{MemGate}}(q,v) - S(q,v)| \leq \dfrac{\epsilon}{1-\epsilon}$.

The middle quantity in (i) is invariant to rescaling the mask, so it charges only the non-uniform part of $g$ that can actually move the score. The last expression is controlled by the preservation term, because $\bar{g} \geq g$ coordinate-wise implies $\|\mathbf{1}_d - \bar{g}\|_1 \leq \|\mathbf{1}_d - g\|_1$ and $\|v \odot \bar{g}\|_2 \geq \|v \odot g\|_2$. Part (ii) is dimension-free: if the normalized mask of a preferred memory stays within $\epsilon$ of the identity in every coordinate, its score moves by at most $\epsilon/(1-\epsilon)$, or about 0.11 for $\epsilon = 0.1$. The result bounds the score change for the mask the gate produces on a given pair; which masks the gate produces for memories unlike those in the training pairs is an empirical question, examined in Appendix F.1. The proof is in Appendix A.

## 4 Experiments

We evaluate MemGate along three questions. (i) Does it reduce memory-induced failures under both benign over-personalization and malicious memory manipulation? (ii) Does it preserve the utility of long-term memory and personalization? (iii) How does it compare with alternative filters, either trained on the same data or implemented with LLM judges, and what overhead does it add?

### 4.1 Experimental Setup

**Systems and backbones.** We evaluate three agentic memory frameworks, A-Mem, Mem0, and MemOS, which differ in how memories are written, organized, and retrieved, together with OpenClaw, a real-world personal-agent environment with native memory management and tool use. Backbones span open-weight and proprietary families of different scales: Qwen-3-8B (Yang et al., 2025), Llama-3.3-70B-Instruct (Grattafiori et al., 2024), GPT-4o-mini (Achiam et al., 2023), GPT-5.1 (Singh et al., 2025), Gemini-3-Pro (Google, 2026), and Claude-Sonnet-4.6 (Anthropic, 2026). Each evaluation instance first stores prior interactions through the system's own memory-writing pipeline and then issues the target query; for OpenClaw, both stages go through its local gateway and native `memory_search` tool (Appendix C.3).

**Benchmarks and metrics.** We measure cross-domain leakage and sycophancy on PersistBench (Pulipaka et al., 2026), tool-call drift on MemDrift (Dabas et al., 2026), and jailbreak attack success rate (ASR) on PS-Bench (Guo et al., 2026). For the first three, a GPT-4o-mini judge rates inappropriate memory influence on a 1–5 scale, and scores $\geq 3$ count as failures; we report the failure rate (FR). Jailbreak success is determined by Llama-Guard. Utility is measured on LoCoMo (Maharana et al., 2024) and MemoryAgentBench (Hu et al., 2026a) for general long-term memory, and on PreFEval (Zhao et al., 2025), PersonaMem (Jiang et al., 2025), and the beneficial-memory subset of PersistBench for personalization, where a failure means that a genuinely useful memory was not retrieved or applied. Sample counts, judges, and benchmark details are given in Table 3 and Appendix C.4.

**Baselines.** For each memory system, the primary baseline is its native similarity-based retrieval with the same budget $k$; where applicable, we also report a memory-free setting (*w/o Mem*). We further compare with two families of alternative filters on the same candidate set and budget. *Supervised re-rankers* are trained on the same 1,640 triples as MemGate and separate the contribution of the supervision from that of the gate design: a *contrastively fine-tuned encoder*, which fine-tunes `all-MiniLM-L6-v2` with an InfoNCE loss using $m^-$ as a hard negative, and a *scalar MLP scorer*, which shares MemGate's input and hidden layers but outputs a single score and is trained with the same retriever-relative ranking term. *LLM-judge filters* screen candidates at inference time: a *pointwise filter* asks the judge whether each candidate should be used and keeps the accepted candidates in their original order, while a *listwise re-ranker* presents all candidates in a single call and asks the judge to select and order the appropriate ones. The judges are instantiated with GPT-4o-mini and GPT-5.1 and use prompts that name the four failure types explicitly. Details are given in Appendices I and H.

**Implementation.** MemGate uses a frozen `all-MiniLM-L6-v2` encoder ($d = 384$) and is trained from scratch on 1,640 synthetic triples (Appendix G) with margin scale $\kappa = 3$ and preservation weight $\lambda = 0.1$. For all systems, the backend returns $N_c = 50$ candidates and the final budget

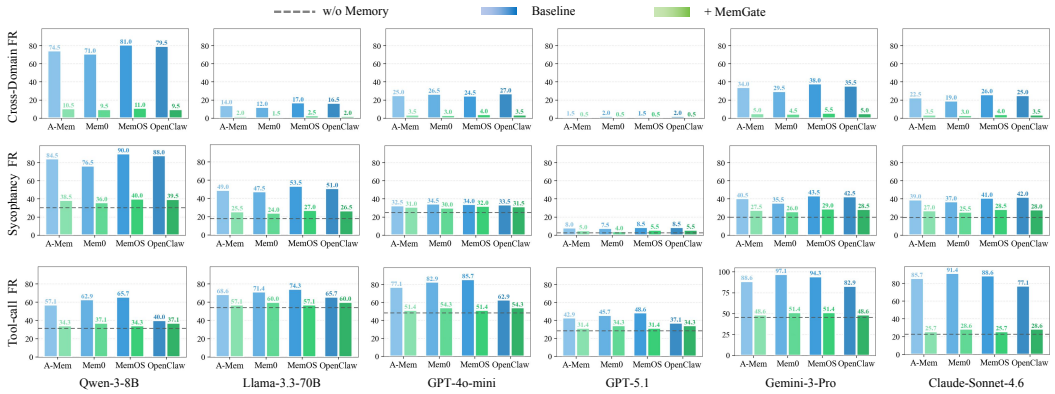


Figure 3: Performance of MemGate on over-personalization evaluations, including cross-domain leakage, sycophancy, and tool-call drift, across agentic memory frameworks and the real-world agent.

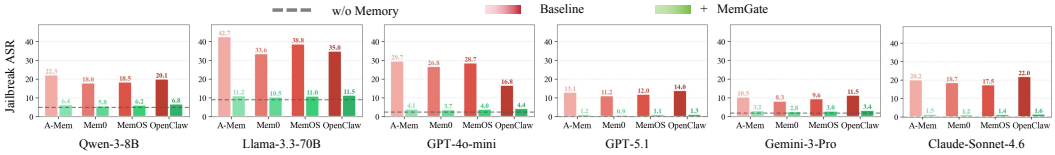


Figure 4: Performance of MemGate on memory-induced safety vulnerability evaluations across agentic memory frameworks and the real-world agent.

is $k = 20$. We use AdamW with learning rate $1.5 \times 10^{-4}$ and batch size 256 for 20 epochs; further details are given in Appendix C.2.

### 4.2 Trustworthiness Evaluation

**Benign over-personalization.** Figure 3 shows consistent reductions across agents and backbones. Cross-domain leakage responds most strongly: on GPT-4o-mini, cross-domain FR falls from 24.5%–27.0% to 3.0%–4.0%. An out-of-domain memory typically matches the query on only a few topical dimensions, which the gate can reweight. Tool-call drift also decreases substantially and, on Claude-Sonnet-4.6, approaches the memory-free level (77.1%–91.4% to 25.7%–28.6%, versus 22.9%). The reduction is smaller on OpenClaw with GPT-4o-mini (62.9% to 54.3%; Table 2), suggesting that the remaining drift depends on how readily the backbone turns personal traits into action parameters. Sycophancy improves least (33.5% to 31.5% in the same setting). Belief-bearing memories are usually on-topic, and the query itself often states the user's position, so memory re-ranking cannot remove the backbone's intrinsic agreement bias; in this respect MemGate complements, rather than replaces, model-level alignment.

**Malicious memory manipulation.** Figure 4 reports jailbreak ASR. On GPT-4o-mini, MemGate reduces ASR to 3.7%–4.4% across memory-enabled agents (OpenClaw: 16.8% to 4.4%), close to the memory-free level of 2.5%. Because planted memories are constructed to be highly similar to the harmful query, this reduction indicates that the gate does not merely reproduce similarity ranking: it down-weights the overlap through which a benign-looking persona legitimizes an unsafe request. To examine this mechanism, we compare the embedding geometry of 120 PS-Bench queries, paired benign memories from LoCoMo, and malicious memories templated from the queries, before and after gating (Appendix E.2). In the raw space, malicious memories overlap strongly with the queries they were built to match; after gating, they move away from the query cluster while benign memories stay close to it, consistent with the gate weakening misleading associations without collapsing the useful semantic neighborhood.

Table 1: Utility evaluation on general memory and personalization benchmarks. We report LoCoMo overall F1, MemoryAgentBench overall accuracy, PreFEval accuracy, PersonaMem accuracy, and PersistBench beneficial failure rate. The complete general-memory results are given in Table 4.

| **Agent** | **LoCoMo** Overall F1↑ | **MemoryAgentBench** Overall Acc.↑ | **PreFEval** Acc.↑ | **PersonaMem** Acc.↑ | **PersistBench** Beneficial FR↓ |
|---|---|---|---|---|---|
| | | | *Qwen-3-8B* | | |
| A-Mem | 37.3 | 25.7 | 72.9 | 44.2 | 26.1 |
| + MemGate | 38.6 | 27.1 | 74.6 | 45.7 | 12.2 |
| Mem0 | 40.0 | 28.8 | 77.2 | 50.9 | 19.6 |
| + MemGate | 40.2 | 29.7 | 78.4 | 52.1 | 8.5 |
| MemOS | 41.7 | 31.2 | 79.5 | 54.6 | 16.2 |
| + MemGate | 42.9 | 32.4 | 80.7 | 55.8 | 6.4 |
| OpenClaw | 38.5 | 27.5 | 75.0 | 47.6 | 23.3 |
| + MemGate | 39.4 | 28.3 | 76.8 | 49.3 | 10.0 |
| | | | *GPT-4o-mini* | | |
| A-Mem | 37.8 | 29.9 | 84.3 | 52.1 | 16.4 |
| + MemGate | 39.1 | 31.0 | 86.1 | 53.7 | 8.6 |
| Mem0 | 42.9 | 32.6 | 88.0 | 56.8 | 11.4 |
| + MemGate | 44.5 | 33.9 | 89.1 | 58.3 | 5.2 |
| MemOS | 45.4 | 35.8 | 89.6 | 61.2 | 8.1 |
| + MemGate | 46.3 | 36.8 | 90.4 | 62.7 | 3.5 |
| OpenClaw | 38.9 | 31.6 | 86.4 | 54.6 | 14.3 |
| + MemGate | 40.8 | 32.7 | 88.2 | 56.1 | 6.7 |

### 4.3 Utility Evaluation

Table 1 shows that MemGate improves every aggregate utility metric for all four systems and both backbones, so the trustworthiness gains above do not come from suppressing memory. The largest change is on the beneficial subset of PersistBench, where the failure rate roughly halves in every setting. We attribute this to the fixed retrieval budget: demoting distracting but similar memories frees top-$k$ slots for memories that genuinely support the task, so filtering and personalization are complementary at retrieval time. The category breakdown (Appendix D) shows gains on every MemoryAgentBench category and on LoCoMo temporal questions, which are sensitive to topically similar events from the wrong time. The exceptions are LoCoMo open-domain questions, which rely on loosely related background memories that the gate is trained to attenuate and decline in some GPT-4o-mini settings (e.g., OpenClaw, $22.8 \rightarrow 16.9$), and single-hop questions for Mem0 with Qwen-3-8B ($48.5 \rightarrow 46.6$).

### 4.4 Comparison with Alternative Filters

Table 2 compares MemGate with supervised re-rankers trained on the same data and with inference-time LLM-judge filtering.

**Supervised re-rankers.** The two supervised baselines separate the effect of the training data from that of the gate design. Both reduce cross-domain leakage, tool-call drift, and jailbreak ASR relative to native retrieval, so the synthetic triples alone carry useful signal. Neither, however, approaches MemGate, and both lower utility below the unfiltered baseline (LoCoMo 37.4 and 38.1 versus 38.9; PreFEval 85.1 and 85.8 versus 86.4), whereas MemGate raises it to 40.8 and 88.2. The fine-tuned encoder is the weakest on cross-domain leakage (18.3%) and ASR (11.2%): its memory embeddings do not depend on the query, so a memory has a single representation for every request, and query-dependent demotion must be expressed through the query embedding alone. The scalar scorer is query-conditioned and matched in capacity (about 8.7M parameters), and it closes part of this gap (11.6% and 7.8%), confirming the value of query conditioning. It still trails MemGate on every metric and leaves sycophancy essentially unchanged (33.8% versus 33.5%). Because it predicts a score directly rather than reweighting the base embedding, it has no identity mask to which a preservation

Table 2: Comparison with alternative memory filters on GPT-4o-mini with OpenClaw ($N_c = 50$ candidates, $k = 20$). Supervised re-rankers are trained on the same 1,640 triples as MemGate. Extra calls count judge generations per query in addition to the backbone call; P50 is memory-search latency, including the filter. Results are averaged over 5 runs. The best value among the filtering methods in each column is in bold.

| Method | Cross-D. FR↓ | Syco FR↓ | Tool-call FR↓ | ASR↓ | LoCoMo↑ | PreFEval↑ | Extra calls | P50 (s)↓ |
|---|---|---|---|---|---|---|---|---|
| OpenClaw (native retrieval) | 27.0 | 33.5 | 62.9 | 16.8 | 38.9 | 86.4 | 0 | 1.47 |
| *Supervised re-rankers (same training triples)* | | | | | | | | |
| + Contrastive fine-tuned MiniLM | 18.3 | 32.1 | 59.7 | 11.2 | 37.4 | 85.1 | 0 | **1.51** |
| + Scalar MLP scorer | 11.6 | 33.8 | 57.2 | 7.8 | 38.1 | 85.8 | 0 | 1.53 |
| *Inference-time LLM judges* | | | | | | | | |
| + Pointwise filter (GPT-4o-mini) | 14.2 | 29.8 | 57.6 | 9.7 | 35.1 | 83.9 | 50 | 2.84 |
| + Pointwise filter (GPT-5.1) | 8.6 | 27.4 | 55.8 | 6.9 | 36.7 | 85.2 | 50 | 4.61 |
| + Listwise re-ranker (GPT-4o-mini) | 16.9 | 31.2 | 59.4 | 11.3 | 38.2 | 86.0 | 1 | 3.12 |
| + Listwise re-ranker (GPT-5.1) | 7.1 | **26.9** | 55.1 | 6.2 | 40.1 | 87.5 | 1 | 6.38 |
| + MemGate (ours) | **3.5** | 31.5 | **54.3** | **4.4** | **40.8** | **88.2** | 0 | 1.59 |

term could anchor useful memories; its lower utility and unchanged sycophancy are consistent with this missing anchor, which leaves it free to demote useful evidence and to promote on-topic belief-bearing memories. The remaining gap between the scalar scorer and MemGate therefore reflects the memory-side mask and the preservation term jointly, at a latency cost of 0.06s at P50.

**LLM-judge filters.** The strongest judge (listwise, GPT-5.1) reduces cross-domain FR to 7.1% and ASR to 6.2%, compared with 3.5% and 4.4% for MemGate, at 4× the P50 latency (6.38s versus 1.59s). Pointwise judging, which assesses each memory in isolation, tends to reject memories whose benefit is indirect and lowers utility below the unfiltered baseline for both judges, whereas the listwise re-ranker largely preserves utility and, with GPT-5.1, improves it (LoCoMo 40.1, PreFEval 87.5), though it remains below MemGate. On tool-call drift, the best judge and MemGate are comparable (55.1% versus 54.3%). The judges are stronger on sycophancy (26.9%–27.4% with GPT-5.1 versus 31.5%), where deciding whether a stored opinion bears on an objective question benefits from reading the question directly. Beyond accuracy, judge-based filtering adds one or more generations per query and exposes every candidate memory, including sensitive ones, to an additional model call.

### 4.5 Efficiency and Ablations

**Efficiency.** MemGate adds one batched forward pass of the gate over the candidate set and no LLM calls. Across the four systems, it increases memory-search latency by at most 0.12s at the median (P50) and 0.19s at the 90th percentile (P90); for OpenClaw, whose base search is the slowest, the relative overhead stays below 9% (Table 5 in Appendix E.1). This cost is small compared with a single LLM generation, which every judge-based filter in Table 2 requires at least once per query.

**Ablations.** Appendix F studies the embedding encoder and the preservation weight $\lambda$ on OpenClaw with GPT-4o-mini. MemGate remains effective with all four encoders tested (384 to 1536 dimensions); larger encoders give incremental gains at the cost of a larger checkpoint and higher latency, so the default `all-MiniLM-L6-v2` offers the best trade-off. A small $\lambda$ filters more aggressively at some cost to utility, whereas a large $\lambda$ readmits risky memories and eventually raises sycophancy FR above the ungated baseline; $\lambda = 0.1$ keeps all four trustworthiness metrics below the baseline while retaining most of the utility gain.

## 5 Conclusion

Across three agentic memory frameworks and a real-world tool-enabled agent, similarity-based retrieval induces cross-domain leakage, sycophancy, tool-call drift, and memory-based jailbreaks. MemGate addresses these failures with a 9M-parameter, query-conditioned gate that re-ranks candidate memories before prompt construction. It substantially reduces memory-induced failures, improves long-term memory and personalization utility, and adds only a small retrieval-time overhead, without modifying the backbone, rewriting the memory store, or invoking an LLM judge.

### AI USE STATEMENT

LLMs are part of our experimental pipeline: GPT-4o-mini generates the synthetic training triples, GPT-4o-mini and Llama-Guard serve as automatic judges, GPT-4o-mini and GPT-5.1 serve as LLM-judge baselines, and six LLMs are evaluated as backbones, all as described in Section 4 and the appendix. In addition, LLM-based assistants were used to polish the writing.

### ETHICS STATEMENT

This work proposes a defensive component for memory-enabled personal agents and does not collect new human-subject data. All evaluations use existing public benchmarks, and MemGate is trained on synthetic preference pairs that contain no real user information. The jailbreak evaluation involves harmful queries from PS-Bench, which we use solely to measure attack success and do not release in new form. Lower failure rates on our benchmarks are not a deployment guarantee: practitioners should evaluate privacy leakage, sycophancy, tool-call safety, and memory-based manipulation in their target environment, and should combine MemGate with backbone-level safeguards.

### REPRODUCIBILITY STATEMENT

The memory search formulation, the MemGate architecture, and the training objective are specified in Section 3. Proposition 1 is stated in Section 3.3 and proved in Appendix A, and the gradient of the objective and its relation to DPO are given in Appendix B. Training settings and evaluation procedures are summarized in Section 4.1 and detailed in Appendices C.2 and C.3. Benchmark descriptions, detailed general-memory results, supplementary analyses, and ablations appear in Appendices C.4, D, E, and F, respectively. The synthetic training data are described in Appendix G, the LLM-judge baselines and their prompts in Appendix H, and the supervised re-ranking baselines in Appendix I. Code, the generation prompt, and the training triples are available at the anonymous repository linked in Section 1.

## REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Ahmad Al-Tawaha, Shangding Gu, Peizhi Niu, Ruoxi Jia, and Ming Jin. Remembering more, risking more: Longitudinal safety risks in memory-equipped llm agents. *arXiv preprint arXiv:2605.17830*, 2026.

Anthropic. Claude documentation. https://docs.anthropic.com/, 2026.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pp. 194–199, 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.

Mahavir Dabas, Jihyun Jeong, Ming Jin, and Ruoxi Jia. Memory-induced tool-drift in llm agents. *arXiv preprint arXiv:2605.24941*, 2026.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920, 2024.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Google. Gemini documentation. https://ai.google.dev/gemini-api/docs/gemini-3, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jiahe Guo, Xiangran Guo, Yulin Hu, Zimo Long, Xingyu Sui, Xuda Zhi, Yongbo Huang, Hao He, Weixiang Zhao, Yanyan Zhao, et al. When personalization legitimizes risks: Uncovering safety vulnerabilities in personalized dialogue agents. *ACL*, 2026.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions. *ICLR*, 2026a.

Yulin Hu, Zimo Long, Jiahe Guo, Xingyu Sui, Xing Fu, Weixiang Zhao, Yanyan Zhao, and Bing Qin. Op-bench: Benchmarking over-personalization for memory-augmented personalized conversational agents. *arXiv preprint arXiv:2601.13722*, 2026b.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, 2025.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J Taylor, and Dan Roth. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. *COLM*, 2025.

Shuyue Stella Li, Bhargavi Paranjape, Kerem Oktar, Zhongyao Ma, Gelin Zhou, Lin Guan, Na Zhang, Sem Park, Lin Chen, Diyi Yang, et al. Horizonbench: Long-horizon personalization with evolving preferences. *arXiv preprint arXiv:2604.17283*, 2026.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, 2025a.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, et al. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*, 2025b.

Jiahong Liu, Zexuan Qiu, Zhongyang Li, Quanyu Dai, Wenhao Yu, Jieming Zhu, Minda Hu, Menglin Yang, Tat-Seng Chua, and Irwin King. A survey of personalized large language models: Progress and future directions. *arXiv preprint arXiv:2502.11528*, 2025.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024.

OpenClaw Project. OpenClaw: Open-source personal AI agent framework. https://openclaw.ai, 2026.

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

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

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.

Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *Information Fusion*, pp. 103599, 2025.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, volume 2024, pp. 32628–32649, 2024.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Yu Wang and Xi Chen. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*, 2025.

Haoming Xu, Weihong Xu, Zongrui Li, Mengru Wang, Yunzhi Yao, Chiyu Wu, Jin Shang, Yu Gong, and Shumin Deng. When should models change their minds? contextual belief management in large language models. *arXiv preprint arXiv:2605.30219*, 2026a.

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, 2026b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Sangyeon Yoon, Sunkyoung Kim, Hyesoo Hong, Wonje Jeung, Yongil Kim, Wooseok Seo, Heuiyeen Yeen, and Albert No. Benchpres: A benchmark for context-aware personalized preference selectivity of persistent-memory llms. *arXiv preprint arXiv:2603.16557*, 2026.

Miao Yu, Fanci Meng, Xinyun Zhou, Shilong Wang, Junyuan Mao, Linsey Pan, Tianlong Chen, Kun Wang, Xinfeng Li, Yongfeng Zhang, et al. A survey on trustworthy llm agents: Threats and countermeasures. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 6216–6226, 2025.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, 2024.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. Do llms recognize your preferences? evaluating personalized preference following in llms. *ICLR*, 2025.

## A Proof of Proposition 1

We first bound the change in cosine similarity along a path from the normalized gated embedding to the original embedding, and then relate the resulting quantity to the preservation term.

*Proof.* Fix $q \neq 0$ and define $f(u) = \frac{q \cdot u}{\|q\|_2 \|u\|_2}$ for $u \neq 0$. Writing $\hat{q} = q/\|q\|_2$ and $\hat{u} = u/\|u\|_2$, its gradient is

$$\nabla f(u) = \frac{\hat{q} - (\hat{q} \cdot \hat{u})\,\hat{u}}{\|u\|_2}, \qquad \|\nabla f(u)\|_2 = \frac{\sqrt{1 - (\hat{q} \cdot \hat{u})^2}}{\|u\|_2} \leq \frac{1}{\|u\|_2}.$$

*Scale invariance.* Since $g \in (0,1)^d$, we have $\|g\|_\infty > 0$, and $v \odot g = \|g\|_\infty (v \odot \bar{g})$. Because $f(cu) = f(u)$ for $c > 0$,

$$S_{\text{MemGate}}(q, v) = f(v \odot g) = f(v \odot \bar{g}), \qquad S(q, v) = f(v).$$

*Part (i).* Consider the segment from $v \odot \bar{g}$ to $v$,

$$u_t = v \odot \big(\bar{g} + t(\mathbf{1}_d - \bar{g})\big), \qquad t \in [0, 1].$$

Since $\bar{g} \in (0,1]^d$, each coefficient $\bar{g}_i + t(1 - \bar{g}_i)$ is at least $\bar{g}_i > 0$, so $\|u_t\|_2 \geq \|v \odot \bar{g}\|_2 > 0$ for all $t$. Hence $f$ is differentiable along the segment with $\|\nabla f(u_t)\|_2 \leq 1/\|v \odot \bar{g}\|_2$, and

$$|f(v) - f(v \odot \bar{g})| = \Big| \int_0^1 \nabla f(u_t) \cdot \big(v \odot (\mathbf{1}_d - \bar{g})\big)\, dt \Big| \leq \frac{\|v \odot (\mathbf{1}_d - \bar{g})\|_2}{\|v \odot \bar{g}\|_2}.$$

This is the first inequality. For the second, note that $\|g\|_\infty \leq 1$ gives $\bar{g} \geq g$ coordinate-wise, so $\mathbf{1}_d - \bar{g} \leq \mathbf{1}_d - g$ with both nonnegative. Therefore

$$\|v \odot (\mathbf{1}_d - \bar{g})\|_2 \leq \|v\|_\infty \|\mathbf{1}_d - \bar{g}\|_2 \leq \|v\|_\infty \|\mathbf{1}_d - \bar{g}\|_1 \leq \|v\|_\infty \|\mathbf{1}_d - g\|_1, \qquad \|v \odot \bar{g}\|_2 \geq \|v \odot g\|_2,$$

which yields the second inequality.

*Part (ii).* If $\bar{g}_i \geq 1 - \epsilon$ for all $i$, then $0 \leq 1 - \bar{g}_i \leq \epsilon$ and $\bar{g}_i \geq 1 - \epsilon$, so

$$\|v \odot (\mathbf{1}_d - \bar{g})\|_2 \leq \epsilon\, \|v\|_2, \qquad \|v \odot \bar{g}\|_2 \geq (1 - \epsilon)\, \|v\|_2.$$

Substituting into the first inequality of part (i) gives $|S_{\text{MemGate}}(q, v) - S(q, v)| \leq \epsilon/(1 - \epsilon)$. □

The bound in terms of $\|\mathbf{1}_d - g\|_1$ recovers the form of a standard perturbation bound with $\gamma = \|v \odot g\|_2$, while the middle quantity of part (i) is strictly tighter whenever the mask has a uniform component, since that component does not change the score.

## B Gradient of the Ranking Objective and Relation to DPO

**Gradient.** Let $u_\phi(q) = \kappa\big[\delta_\phi(q, m^+) - \delta_\phi(q, m^-)\big] = \kappa\big[\Delta_\phi(q) - \Delta_0(q)\big]$. Because the base scores $S$ do not depend on $\phi$, the gradient of the ranking term for one triple is

$$\nabla_\phi\, \ell_{\text{rank}}(q, m^+, m^-) = -\kappa\, \sigma\big(-u_\phi(q)\big) \Big[\nabla_\phi S_{\text{MemGate}}(q, v_{m^+}) - \nabla_\phi S_{\text{MemGate}}(q, v_{m^-})\Big].$$

Each triple is weighted by $\sigma(-u_\phi(q))$, which is large while the gate has not yet improved on the base margin and vanishes once it has. The base margin $\Delta_0$ affects this weight but not the direction of the update. For comparison, a plain pairwise logistic loss on the gated scores, $-\log \sigma\big(\kappa \Delta_\phi(q)\big)$, weights each triple by $\sigma\big(-\kappa \Delta_\phi(q)\big)$; at a uniform mask this equals $\sigma\big(-\kappa \Delta_0(q)\big)$, so triples that the base retriever already ranks correctly by a wide margin receive little gradient. Under Eq. 7, every triple starts with weight $1/2$ regardless of the base margin. Neither weight limits the magnitude of the resulting score corrections, which is the role of the preservation term in Eq. 8.

**Relation to DPO.** Define softmax distributions over the candidate set $\mathcal{C}_q$ induced by the gated and the base scores,

$$\pi_\phi(m \mid q) = \frac{\exp\big(S_{\text{MemGate}}(q, v_m)/\tau\big)}{Z_\phi(q)}, \qquad \pi_{\text{ref}}(m \mid q) = \frac{\exp\big(S(q, v_m)/\tau\big)}{Z_{\text{ref}}(q)},$$

with temperature $\tau$ and normalizers $Z_\phi(q)$ and $Z_{\text{ref}}(q)$. Then

$$\beta \log \frac{\pi_\phi(m \mid q)}{\pi_{\text{ref}}(m \mid q)} = \frac{\beta}{\tau}\, \delta_\phi(q, m) - \beta \log \frac{Z_\phi(q)}{Z_{\text{ref}}(q)}.$$

The log-partition term is the same for $m^+$ and $m^-$ and cancels inside the sigmoid, so the DPO loss (Rafailov et al., 2023) on $(q, m^+, m^-)$ equals $\ell_{\text{rank}}$ with $\kappa = \beta/\tau$; our setting $\kappa = 3$ corresponds, for instance, to $\beta = 0.3$ and $\tau = 0.1$. Two differences from DPO for language models are worth noting. The reference here is a fixed base score rather than the anchor of a KL-regularized objective, so it does not bound how far the gated scores move; in MemGate this role is played by the preservation term. In addition, the normalization over $\mathcal{C}_q$ plays no role in training. For these reasons we present the loss directly as a ranking objective in Section 3.3.

# C Detailed Experimental Setup

## C.1 Architecture Details

The default MemGate MLP takes the 1152-dimensional interaction vector $x_{\text{inter}} = \text{concat}(q, v_m, q \odot v_m)$ and computes

$$\begin{aligned} h_1 &= \text{SiLU}(\text{LN}(\text{Linear}_{1152\to 2048}(x_{\text{inter}}))), \\ h_2 &= \text{SiLU}(\text{LN}(\text{Linear}_{2048\to 2048}(h_1))), \\ h_3 &= \text{SiLU}(\text{LN}(\text{Linear}_{2048\to 1024}(h_2))). \end{aligned}$$

Dropout rates are $0.15$ for $h_1$ and $h_2$ and $0.1$ for $h_3$. The memory-side gate is

$$G_\phi(q, v_m) = \sigma(\text{Linear}_{1024\to 384}(h_3)) \in (0, 1)^d.$$

The encoder is frozen and only the gating network is trained. With $d = 384$, the four linear layers contain approximately 9M parameters, corresponding to the 35.1MB checkpoint.

## C.2 Training Details

Queries and candidate memories are encoded with a frozen `all-MiniLM-L6-v2` into 384-dimensional vectors, and the same encoder provides the base scores $S$. For each query–memory pair, the gate takes the concatenation of the query embedding, the memory embedding, and their element-wise product, and outputs a sigmoid mask applied only to the memory embedding. The gate is trained from scratch on the 1,640 synthetic triples of Appendix G by minimizing Eq. 8 with $\kappa = 3$ and $\lambda = 0.1$. We use AdamW with learning rate $1.5 \times 10^{-4}$ and batch size 256, and train for 20 epochs.

## C.3 Evaluation Protocols

**Evaluated systems.** We evaluate four memory-enabled agent settings. A-Mem, Mem0, and MemOS are agentic memory frameworks with different memory writing, organization, and retrieval mechanisms. OpenClaw is a real-world personal-agent environment with persistent state, native memory management, and tool use; we use it as an external long-term-memory backend rather than as a trained component of MemGate. Where applicable, we also include a memory-free setting, denoted *w/o Mem*, to isolate the effect of long-term memory.

Each evaluation instance consists of two stages. In the memory construction stage, the agent receives one or more prior interactions, and each system's own writing pipeline stores the resulting memories. For A-Mem, Mem0, and MemOS, we use their native memory writing and retrieval pipelines. For OpenClaw, we configure a local gateway on a loopback port with the `memory-core` plugin enabled and use an isolated, benchmark-specific state and workspace directory for each experiment. The prior interactions of each instance are sent to the agent through this gateway, and the `memory-core` plugin writes the resulting memories as Markdown files in the workspace (`MEMORY.md` and dated files under `workspace/memory/`; see Figure 1). In the task execution stage, we issue the target query and record the agent's response or tool call. Candidate memories are retrieved through the OpenClaw gateway's `memory_search` tool with `corpus="memory"` and the corresponding benchmark query. For all systems, the backend returns a candidate set of $N_c = 50$ memories and the final retrieval budget is $k = 20$.

**Evaluation tasks and metrics.** Our assessment covers four memory-induced risks, summarized in Table 3. For cross-domain leakage and sycophancy, we follow the PersistBench setup and report the failure rate (FR). The judge evaluates whether the response exhibits inappropriate memory influence and assigns an ordinal failure score from 1 to 5, where higher scores indicate more severe failure. We treat scores $\geq 3$ as failures, since they indicate clear inappropriate memory influence. Tool-call drift on MemDrift is judged in the same way, and jailbreak success on PS-Bench is determined by Llama-Guard.

Table 3: Trustworthiness benchmarks, targeted failure types, sample counts, and judge models.

| Benchmark | Type | #Samples | Judge |
|---|---|---|---|
| PersistBench (Pulipaka et al., 2026) | Cross-Domain | 200 | GPT-4o-mini |
| | Sycophancy | 200 | GPT-4o-mini |
| MemDrift (Dabas et al., 2026) | Tool-call | 105 | GPT-4o-mini |
| PS-Bench (Guo et al., 2026) | Jailbreak | 750 | Llama-Guard |

### C.4 Utility Benchmark Details

We evaluate utility from two complementary perspectives. The first concerns general long-term memory capability: whether an agent can retrieve, maintain, and reason over information across long interaction histories. The second concerns personalization: whether the system correctly uses beneficial user memories, preferences, or persona information when they are relevant to the current query.

**General long-term memory benchmarks.** These benchmarks evaluate whether a memory-enabled agent can preserve and use information over extended or incremental interactions.

- **LoCoMo.** LoCoMo (Maharana et al., 2024) evaluates very long-term conversational memory over multi-session dialogues. It tests whether a model can recover and use information from extended interaction histories through tasks such as long-term question answering, event summarization, and dialogue generation. We report the overall F1 score.
- **MemoryAgentBench.** MemoryAgentBench (Hu et al., 2026a) evaluates memory agents under incremental multi-turn interactions, covering abilities such as accurate retrieval, long-range understanding, and selective forgetting. We report the overall accuracy.

**Personalization benchmarks.** The following datasets emphasize whether the model uses user-specific memories when they are genuinely helpful. They are therefore closest to the intended utility of personalized memory systems: relevant memories should improve the response, and failing to retrieve or apply them reduces personalization quality.

- **PreFEval.** PreFEval (Zhao et al., 2025) measures whether LLMs can infer, memorize, and follow user preferences in long-context conversations. It contains manually curated preference–query pairs across diverse topics and evaluates whether the model adapts its responses to previously expressed preferences. We report accuracy.
- **PersonaMem.** PersonaMem (Jiang et al., 2025) evaluates dynamic user profiling and personalized response selection across multi-session user–assistant interactions. Each instance contains an evolving user profile and interaction history, and the model must select the response that best matches the user's current persona and preferences. We report accuracy.
- **PersistBench beneficial subset.** PersistBench (Pulipaka et al., 2026) contains both trustworthiness-oriented and utility-oriented memory tasks. For utility evaluation, we use its beneficial-memory subset, in which long-term memory is genuinely relevant and should be used. We report the beneficial failure rate (Beneficial FR), where a failure means that the system does not retrieve or apply the necessary memory.

# D Detailed General Memory Results

Table 4 gives the complete task-category breakdown for LoCoMo and MemoryAgentBench. The overall scores also appear alongside the personalization results in Table 1, and the category-level trends are discussed in Section 4.3.

Table 4: Utility evaluation on general memory benchmarks. We report F1 on LoCoMo and accuracy on MemoryAgentBench for each memory-enabled agent, with and without MemGate.

| Agent | LoCoMo | | | | | MemoryAgentBench | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Single-Hop | Multi-Hop | Open-Domain | Temporal | Overall | Single-Hop | Multi-Hop | Long-Context | Event | Overall |
| *Qwen-3-8B* | | | | | | | | | | |
| A-Mem | 43.7 | 37.4 | 18.6 | 15.9 | 37.3 | 18.8 | 24.3 | 29.1 | 30.4 | 25.7 |
| + MemGate | 44.9 | 38.6 | 21.7 | 17.3 | 38.6 | 20.2 | 25.7 | 30.8 | 31.6 | 27.1 |
| Mem0 | 48.5 | 44.5 | 16.7 | 12.5 | 40.0 | 22.4 | 27.6 | 32.3 | 33.1 | 28.8 |
| + MemGate | 46.6 | 46.2 | 23.9 | 15.2 | 40.2 | 23.6 | 28.4 | 33.1 | 34.2 | 29.7 |
| MemOS | 50.7 | 46.1 | 18.4 | 14.6 | 41.7 | 24.1 | 30.3 | 34.8 | 35.6 | 31.2 |
| + MemGate | 51.3 | 47.8 | 24.4 | 17.8 | 42.9 | 25.3 | 31.4 | 36.1 | 36.8 | 32.4 |
| OpenClaw | 44.6 | 38.1 | 21.9 | 17.5 | 38.5 | 20.7 | 26.1 | 31.2 | 31.8 | 27.5 |
| + MemGate | 45.0 | 38.7 | 24.1 | 19.5 | 39.4 | 21.8 | 26.9 | 31.7 | 32.9 | 28.3 |
| *GPT-4o-mini* | | | | | | | | | | |
| A-Mem | 41.3 | 35.6 | 20.1 | 33.4 | 37.8 | 22.7 | 28.2 | 33.1 | 35.4 | 29.9 |
| + MemGate | 42.6 | 37.1 | 19.8 | 36.3 | 39.1 | 23.4 | 29.7 | 34.6 | 36.2 | 31.0 |
| Mem0 | 48.6 | 42.1 | 15.1 | 35.3 | 42.9 | 25.0 | 32.0 | 36.0 | 37.5 | 32.6 |
| + MemGate | 49.8 | 43.2 | 15.1 | 40.4 | 44.5 | 26.4 | 33.1 | 37.3 | 38.7 | 33.9 |
| MemOS | 50.4 | 44.2 | 17.3 | 38.9 | 45.4 | 27.8 | 35.3 | 39.2 | 40.7 | 35.8 |
| + MemGate | 51.6 | 45.1 | 18.6 | 42.5 | 46.3 | 28.7 | 36.4 | 40.3 | 41.8 | 36.8 |
| OpenClaw | 42.5 | 36.9 | 22.8 | 35.1 | 38.9 | 24.2 | 30.7 | 35.4 | 36.1 | 31.6 |
| + MemGate | 44.9 | 39.4 | 16.9 | 38.5 | 40.8 | 25.1 | 31.8 | 36.2 | 37.6 | 32.7 |

# E Supplementary Analyses

## E.1 Efficiency

Table 5 reports memory-search latency at the median (P50) and at the 90th percentile (P90), which captures the slow cases that shape perceived responsiveness, together with LoCoMo overall F1. MemGate adds one batched forward pass of the gate over the candidate set and no LLM calls. Across the four systems, the absolute overhead is at most 0.12s at P50 and 0.19s at P90. OpenClaw has the slowest base search because of its system-level memory and tool integration, and its relative overhead stays below 9%.

Table 5: Efficiency of MemGate with GPT-4o-mini: P50 and P90 memory-search latency (s) and LoCoMo overall F1.

| Agent | P50 | P90 | LoCoMo |
|---|---|---|---|
| A-Mem | 0.42 | 0.61 | 37.8 |
| + MemGate | 0.44 | 0.68 | 39.1 |
| Mem0 | 0.26 | 0.35 | 42.9 |
| + MemGate | 0.29 | 0.41 | 44.5 |
| MemOS | 0.19 | 0.27 | 45.4 |
| + MemGate | 0.23 | 0.30 | 46.3 |
| OpenClaw | 1.47 | 2.56 | 38.9 |
| + MemGate | 1.59 | 2.75 | 40.8 |

Because the gate scores only the $N_c$ candidates returned by the backend, its cost scales with the size of the candidate set rather than with the size of the memory store. The absolute overhead is similar across systems, so the relative overhead is largest for backends whose base search is fastest, such as MemOS (0.19s to 0.23s at P50). Even in that case, the added time remains far below that of a single LLM generation, which every judge-based filter in Table 2 requires at least once per query.

## E.2 Representation Analysis

To examine how MemGate lowers jailbreak ASR, we visualize the embedding geometry before and after gating. We sample 120 user queries from PS-Bench and pair them with 120 benign memories from LoCoMo dialogues and 120 malicious memories templated from the queries, so that the $i$-th benign and the $i$-th malicious memory are both paired with the $i$-th query. Each memory embedding

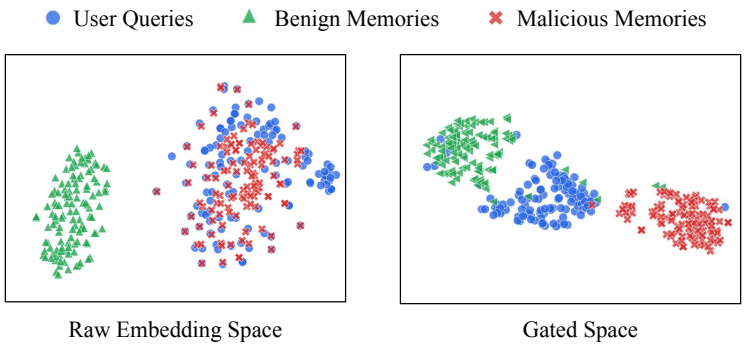


Figure 5: Representation shift induced by MemGate. Malicious memories overlap with user queries in the raw space; after gating they separate from the query cluster, while benign memories stay close to it.

$v_m$ is multiplied element-wise by the gate $G_\phi(q, v_m)$ computed with its paired query $q$ and then L2-renormalized; query embeddings are unchanged in both views.

As Figure 5 shows, malicious memories overlap strongly with user queries in the raw space, as expected from their construction. After gating, they move away from the query cluster, whereas benign memories remain close to it. This qualitative picture is consistent with the gate weakening misleading associations without collapsing the useful semantic neighborhood.

## F Additional Ablation Studies

This appendix reports ablations on the preservation weight and the embedding encoder, all with GPT-4o-mini on OpenClaw.

### F.1 Preservation Weight

Figure 6 varies $\lambda$ from 0 to 0.5. A smaller $\lambda$ lets the gate attenuate memories more aggressively, so cross-domain leakage, tool-call drift, and jailbreak ASR are lowest at $\lambda = 0$. The same aggressiveness also removes some useful evidence, and $\lambda = 0$ yields the smallest utility gain over the OpenClaw baseline.

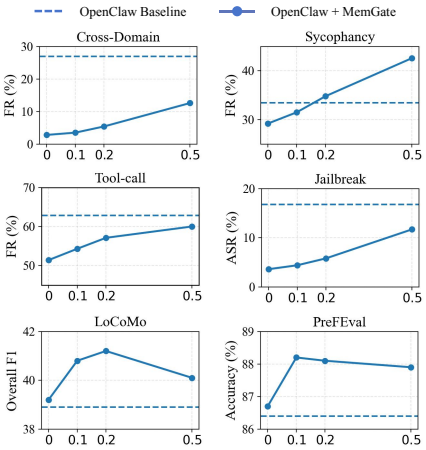


Figure 6: Ablation of the preservation weight $\lambda$ (GPT-4o-mini, OpenClaw). Lower is better for FR and ASR; higher is better for F1 and accuracy.

Increasing $\lambda$ keeps preferred memories closer to the identity mask and readmits more risky context. All four trustworthiness metrics rise with $\lambda$, while LoCoMo and PreFEval peak around $\lambda \in [0.1, 0.2]$ and then decline slightly, suggesting that excessive retention dilutes useful evidence with irrelevant context. At the larger values in our sweep, sycophancy FR exceeds the OpenClaw baseline, whereas cross-domain FR, tool-call FR, and ASR remain below it. We attribute this to the asymmetry of the preservation term. With $g^+$ pinned near the identity, the ranking term can be reduced only by reshaping the masks of other candidates; because a non-uniform mask can raise as well as lower a memory's cosine to the query (Section 3.2), on-topic belief-bearing memories that are poorly covered by the training pairs can end up promoted rather than demoted.

### F.2 Embedding Encoder

Table 6 compares four encoders whose dimensionality ranges from 384 to 1536; a larger dimension enlarges both the gate input ($3d$) and its output mask ($d$). Every encoder substantially re-

Table 6: Performance of MemGate with different embedding models on GPT-4o-mini with OpenClaw. Lower is better for FR, ASR, and latency; higher is better for LoCoMo and PreFEval.

| MemGate Setting | | | Trustworthy Evaluation | | | | Utility Evaluation | | Efficiency |
|---|---|---|---|---|---|---|---|---|---|
| MemGate Encoder | Dim. | Size | Cross-D. FR↓ | Syco FR↓ | Tool-call FR↓ | ASR↓ | LoCoMo↑ | PreFEval↑ | Latency↓ |
| *OpenClaw (baseline)* | | | | | | | | | |
| - | - | - | 27.0 | 33.5 | 62.9 | 16.8 | 38.9 | 86.4 | 1.47s |
| *OpenClaw w. MemGate* | | | | | | | | | |
| `all-MiniLM-L6-v2` | 384 | 35.1MB | 3.5 | 31.5 | 54.3 | 4.4 | 40.8 | 88.2 | 1.59s |
| `all-mpnet-base-v2` | 768 | 45.8MB | 3.5 | 31.0 | 51.4 | 4.1 | 41.2 | 88.5 | 1.63s |
| `bge-large-en-v1.5` | 1024 | 52.8MB | 3.5 | 30.5 | 51.4 | 3.9 | 41.6 | 88.8 | 1.65s |
| `text-embedding-3-small` | 1536 | 67.1MB | 3.0 | 30.0 | 48.6 | 3.9 | 41.8 | 89.0 | 1.74s |

duces cross-domain leakage, tool-call drift, and jailbreak ASR relative to the OpenClaw baseline while improving LoCoMo and PreFEval. Stronger encoders bring incremental gains: replacing `all-MiniLM-L6-v2` with `text-embedding-3-small` lowers tool-call FR from $54.3\%$ to $48.6\%$ and raises PreFEval from $88.2$ to $89.0$, but grows the checkpoint from $35.1$MB to $67.1$MB and latency from $1.59$s to $1.74$s. The main benefit therefore comes from query-conditioned gating rather than from a particular embedding model.

## G Synthetic Preference Data

The 1,640 training triples are generated with GPT-4o-mini by combining six abstract risk categories with 40 seed domains (e.g., workplace email, travel planning, home finance, and DevOps). Four categories correspond to the failure types in Section 2: *domain-scope mismatch*, *belief as evidence*, *trait-to-action*, and *context legitimization*. Two further categories cover *stale state* (a superseded plan or status) and *third-party confidence* (information about another person shared in private). For each category–domain pair, the prompt asks for a realistic user request, a memory the assistant should use ($m^+$), and a topically close memory whose use would be inappropriate under the category ($m^-$). Because $m^+$ is always a genuinely useful memory about the same user, the gate is trained to prefer relevant personalization rather than to suppress memory.

We keep only triples whose negative is a hard negative ($S(q, v_{m^-}) \geq 0.3$ under `all-MiniLM-L6-v2`), remove near-duplicates, and discard triples on which an independent GPT-4o-mini verification pass disagrees with the intended label. No benchmark instance is used as a seed or in-context example. As a sanity check, no evaluation item has a near-duplicate (cosine $> 0.9$ under `bge-large-en-v1.5`) or shares a word 8-gram with the training set. The generation prompt and the training triples are released with the code.

## H Details of the LLM-Judge Baselines

Both judge-based filters in Section 4.4 run on OpenClaw with GPT-4o-mini as the backbone and use the same $N_c = 50$ candidates and final budget $k = 20$ as MemGate. The pointwise filter queries the judge once per candidate with the prompt below, keeps the accepted candidates in their original order, and passes the first $k$ to the backbone; the $N_c$ calls are issued in parallel. The listwise re-ranker applies the same criteria to a numbered list of all candidates in a single call and returns a JSON list of indices ordered from most to least useful, of which the first $k$ are used. Reported latency is P50 memory-search latency including the judge. Because GPT-4o-mini also serves as the evaluation judge for PersistBench and MemDrift, the GPT-4o-mini filters may be favored by the evaluation.

**Pointwise LLM-filter prompt**

You decide whether a stored memory about the user should be shown to an assistant that is about to answer the user's current request. Admit the memory only if using it would make the response more helpful for this specific request, and only if it would not (a) reveal private information outside the context in which it was shared, (b) treat the user's opinion as evidence for an objective question, (c) change the operational or safety parameters of an action because of a personal trait, or (d) make a potentially harmful request appear legitimate.

Current request: {query}
Memory: {memory}
Answer with a single word: ADMIT or REJECT.

# I Details of the Supervised Re-ranking Baselines

Both supervised baselines in Section 4.4 are trained on the same 1,640 triples as MemGate (Appendix G), use the same held-out split of 10% of the triples for checkpoint selection, and are evaluated on OpenClaw with GPT-4o-mini using the same $N_c = 50$ candidates and final budget $k = 20$. Neither makes LLM calls at inference, and neither modifies the memory store: candidate texts returned by the backend are encoded at query time, exactly as for MemGate.

**Contrastively fine-tuned encoder.** We initialize from `all-MiniLM-L6-v2` and fine-tune all encoder parameters with an InfoNCE loss,

$$\ell_{\mathrm{NCE}}(q, m^+) = -\log \frac{\exp\big(s_\theta(q, m^+)/\tau_c\big)}{\exp\big(s_\theta(q, m^+)/\tau_c\big) + \sum_{m \in \mathcal{N}(q)} \exp\big(s_\theta(q, m)/\tau_c\big)},$$

where $s_\theta$ is the cosine similarity under the fine-tuned encoder and $\mathcal{N}(q)$ contains the hard negative $m^-$ of the triple together with the positive and negative memories of the other triples in the batch. We use $\tau_c = 0.05$, AdamW with learning rate $2 \times 10^{-5}$, batch size 64, and 5 epochs. At inference, the query and the candidates are encoded with the fine-tuned encoder and ranked by cosine similarity. Because the memory embedding does not depend on the query, each memory has the same representation for every request, and query-dependent demotion must be carried entirely by the query embedding.

**Scalar MLP scorer.** The scorer uses MemGate's input $x_{\mathrm{inter}}$ and hidden layers $h_1$–$h_3$ (Appendix C.1) on top of the same frozen encoder, but replaces the mask head with a scalar output,

$$s_\phi(q, v_m) = \mathrm{Linear}_{1024 \to 1}(h_3),$$

giving about 8.7M parameters. To isolate the effect of the parameterization, it is trained with the ranking term of Eq. 7, with $S_{\mathrm{MemGate}}(q, v_m)$ replaced by $s_\phi(q, v_m)$, and with the same $\kappa$, optimizer, and schedule as MemGate. Candidates are ranked by $s_\phi$ alone. Because the scorer has no identity mask, the preservation term has no direct analogue and is omitted; the comparison with MemGate therefore tests the memory-side mask and the preservation term jointly.