SALIMORY: ORCHESTRATING COGNITIVE MEMORY FOR CONVERSATIONAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recall and personalization, enabled by memory, are pivotal to a lifelong conversational agent. However, simply expanding context windows with raw retrieval degrades reasoning quality, while training memory agents via standard reinforcement learning creates a severe credit assignment bottleneck in a multi-stage pipeline. To solve this, we introduce SALIMORY, a framework that learns to manage and utilize a cognitively-structured memory—spanning user facts, preferences, and working memory. By introducing novel stage-wise process rewards and reward-decomposed contrastive refinement, SALIMORY provides isolated supervision for distinct memory operations (selective filtering, consolidation, and cue-driven recall) end-to-end. Across four datasets, SALIMORY records the highest Good Personalization rate at 39.8% and the best QA accuracy at 72.9%, while consistently producing the least vague and most factual memories, keeping vagueness under 20% and factuality above 89%.

1 INTRODUCTION

The evolution of conversational AI is rapidly shifting from isolated, single-turn interactions toward lifelong digital companions. As AI agents become embedded in daily life, maintaining persistent context beyond the limits of standard LLM context windows has become a necessity (Xu et al., 2022; Bae et al., 2022). However, simply attaching an external memory storage is not enough: users' interaction histories are often vast, repetitive, and dominated by trivia. The real challenge is *management*—deciding *what to remember*, *how to organize memories*, and *what to recall for a given context*. Two main paradigms have emerged, but both fall short. One extreme is to store all past interactions and apply Retrieval-Augmented Generation (RAG) to surface relevant context when a new question arrives. As the memory accumulates, however, the retrieval results tend to become noisy and repetitive (Borgeaud et al., 2022; Zhong et al., 2024; Lee et al., 2024). Another extreme is to actively compress and curate long-term memories, but neither prompting-based (Li et al., 2024; Xu et al., 2025b) nor reinforcement-learning–based (Yan et al., 2025; Wang et al., 2025) variants have proved to maintain compressed memories to enable reliable capture of user preferences while preserving salient but nuanced details required for future conversations. Our error analysis on the LoCoMo benchmark (Maharana et al., 2024) reveals that in state-of-the-art solutions imprecise or incomplete memory can block well-personalized answers on up to 52% of questions.

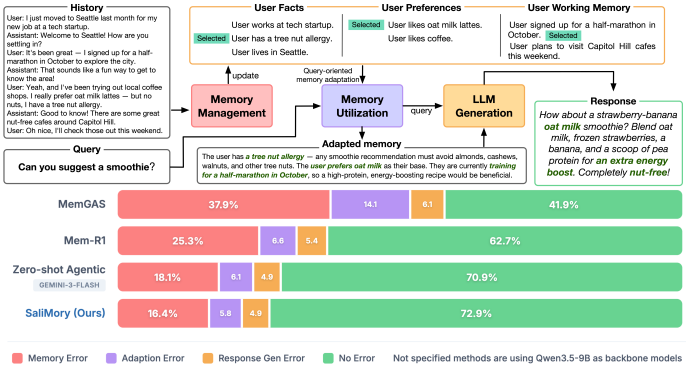

Figure 1: **Gated error attribution on LoCoMo.** We trace each incorrect response to its earliest point of failure in the pipeline (top). Memory errors—where salient information is lost or distorted during formation—dominate across baselines (e.g., 37.9% for MemGAS), vastly exceeding profile generation and response generation errors combined.

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.

How do human beings cope with a lifetime of experiences without being overwhelmed? Cognitive science suggests we do not try to remember everything (Atkinson & Shiffrin, 1968; Baddeley, 2000): *selective attention* filters what gets encoded, as recent traces remain active in *working memory* to shape ongoing decisions (Baddeley & Hitch, 1974). What survives long-term is structurally organized: enduring *facts* serve as hard constraints on what is true, while value-laden *preferences* act as soft biases on what feels right (Tulving, 1972). Consolidation itself is shaped by downstream outcomes via dopaminergic reward (Shohamy & Adcock, 2010; Lisman et al., 2011).

Guided by these principles, we propose **SALIMORY**, an agentic memory-management framework built around three complementary stores: a *factual snapshot* of verifiable facts about the user, a set of *long-term preferences* capturing subjective tastes, and a *short-term working memory* of recent details the user likely still has in mind. The factual snapshot supplies hard constraints; preferences supply soft criteria; working memory surfaces emerging interests worth revisiting. A single memory-management module operates on these stores in three roles: (i) *selective attention*: judging which turns are salient enough to record; (ii) *consolidation*: updating and forgetting memory; and (iii) *cue-driven utilization*: retrieving and applying relevant memories at inference.

Reinforcement learning (RL) for training a learnable memory has been extensively studied (Zhou et al., 2025; Zhang et al., 2026b; Yan et al., 2025). However, existing reward designs are too sparse and distant to supervise intermediate memory decisions. We introduce two complementary mechanisms: The **stage-wise process reward** scores not only final-answer correctness and personalization, but also the quality of the resulting memory and the soundness of each intermediate decision (saliency, utilization). A **reward-decomposed contrastive refinement** step then amplifies the memory-management signal: within each Group Relative Policy Optimization (GRPO) (Shao et al., 2024) rollout batch, we construct role-specific preference pairs that upweight memory decisions in the overall objective. Together, SALIMORY enables a memory agent to memorize and utilize cognitive memory wisely in a human-like manner. In summary, this paper makes the following contributions.

- **Cognitively inspired memory architecture.** We separate factual snapshots, subjective preferences, and short-term working memory, and define three complementary memory-management roles—saliency filter, memory booster, and memory utilizer—that together ground long-horizon conversations.
- **Memory-anchored RL.** We propose a process reward system that tracks memories in each stage and design reward-decomposed contrastive refinements over GRPO rollouts, assuring high-quality memory formation and end response generation.
- **New benchmark and evaluation protocol.** To fully evaluate memory's impact on conversational agents, we introduce a new **LoCoMo-P13n** benchmark with 2k personalizable queries. We also introduce a two-level evaluation protocol that jointly stresses end performance and intermediate memory quality. Empirically, a 9B SALIMORY achieves the highest Good Personalization rate and QA accuracy on four datasets, backed by the least vague and most factual memories throughout.

2 RELATED WORK

Memory-Augmented LLM. Equipping LLMs with persistent context traditionally follows two paradigms. Parametric Memory updates model weights via fine-tuning (e.g., FireAct (Chen et al., 2023), AgentLumos (Yin et al., 2024)) or soft parameters (SELF-PARAM (Wang et al., 2024)), but suffers from catastrophic forgetting and poor generalization to unseen queries. RAG-based Memory abstracts (Borgeaud et al., 2022; Lee et al., 2024) experiences into external datastores, using semantic search to fetch relevant context. However, it struggles to reliably distinguish critical objective facts and connections from transient remarks.

Conversational Memory Structures. To overcome RAG's limitations, agentic systems organize context into distinct topologies. Linear memory treats history sequentially; MemGPT (Packer et al., 2023) manages context using OS-like FIFO queues, while MemoryBank (Zhong et al., 2024) decays older interactions. Layered memory stratifies information to prioritize relevance, with MemoryOS (Kang et al., 2025; Li et al., 2025) segregating short, mid, and long-term stores based on access frequency and user's persona. Moreover, tree and graph-based memories map relational dependencies, with A-Mem (Xu et al., 2025b) constructing dynamic networks, AssoMem (Zhang et al., 2026a) mimicking human associative memory for dense searches, and Mem0 (Chhikara et al.,

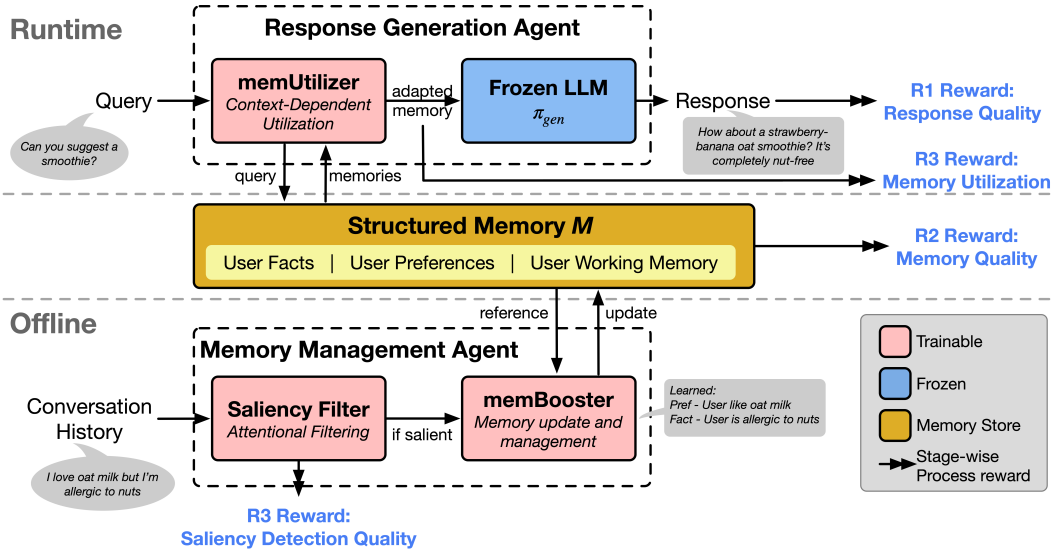

Figure 2: SALIMORY learns to manage and utilize cognitive memory for conversational agents.

2025) utilizing entity-centric graphs. While these topologies improve organization, they rely on fragile heuristics and lack considerations on memory itself.

Reinforcement Learning for Memory Management. Recent works increasingly formulate memory management as an RL problem. MEM1 (Zhou et al., 2025; Zhang et al., 2025) uses RL for learning to manage a memory, but its monolithic memory failed to take downstream task performance into consideration. Conversely, MemRL (Zhang et al., 2026b) applies runtime RL to update retrieval Q-values but ignores memory creation quality. Memory-R1 (Yan et al., 2025) and Mem-α (Wang et al., 2025) capture connections between the memory creation and downstream tasks, which train agents to execute explicit memory management and utilization. While pioneering, their monolithic rewards obscure where failures stem from in a multi-stage pipeline (Tan et al., 2026).

Different from these previous efforts, SALIMORY connects downstream task performance and memory quality jointly and provides novel fine-grained credit assignments required to optimize a structured, multi-stage memory system end-to-end via RL algorithm.

3 METHODOLOGY

We present SALIMORY, an end-to-end agentic memory framework designed to coordinate and optimize conversational memory for downstream response generation.

3.1 PROBLEM DEFINITION AND MEMORY BANK

In *long-horizon conversational* settings, an agent receives all conversation history $\mathcal{H}$ and a current user query q. The goal is to produce a *contextualized* and *personalized* answer $\hat{a}$ that satisfies two criteria: accurately answering the query, and seamlessly grounding in the personal history.

Because $\mathcal{H}$ grows unboundedly and is dominated by trivia, we do not condition on it directly. Instead, we maintain a dynamically evolving *memory bank* $\mathcal{M}$, derived from $\mathcal{H}$, and reformulate the task as $\hat{a} = \pi_{\text{gen}}(q, \mathcal{M})$. Inspired by human memory systems, $\mathcal{M}$ is not a flat buffer but three dedicated stores, $\mathcal{M} = (\bar{F}, \bar{P}, \bar{W})$: (i) *factual bank* $\bar{F}$: objective facts that are verifiable (e.g., being lactose intolerant); (ii) *preference bank* $\bar{P}$: subjective tastes and leanings that act as soft criteria for stylistic tailoring (e.g., preference for oat milk); (iii) *working memory* $\bar{W}$: recent salient turns the user likely still has in mind (e.g., asking about organic food this morning). Separating long-term profile from short-term context, and objective facts from subjective preferences, allows the agent to differentiate, yielding sharper personalization than a flat context window allows.

Given this structured memory, the core challenge shifts from storage to *dynamic management*. An effective memory manager must autonomously resolve three sequential decisions:

(i) *What enters memory?* Act as an attentional gate, deciding which turns carry salient information worth extracting and which to discard.

(ii) *How is memory updated?* Route each extraction to the correct store ($\bar{F}$, $\bar{P}$, or $\bar{W}$), reconcile with existing entries, and retire stale items from working memory.

(iii) *How is memory used?* At query time, synthesize relevant entries across the three stores into a query-adapted profile that conditions the generator.

3.2 THE SALIMORY ARCHITECTURE AND RL FRAMEWORK

To separate the cognitive load of memory management from conversational reasoning for latency optimization, the SALIMORY architecture (Figure 2) contains two distinct computational stages: an *offline stage* that builds and maintains the structured memory, and a *runtime stage* that leverages it.

The Offline Stage processes each session of conversations asynchronously to produce an updated memory bank $\mathcal{M}$. First, the *Saliency Filter* drops transient noise. Second, the *memBooster* processes salient turns, incrementally updates the factual bank $\bar{F}$ and the preference bank $\bar{P}$, and maintains the working memory bank $\bar{W}$ within a sliding time window.

The Runtime Stage executes when a new user query q arrives. The *memUtilizer* finds relevant memories from the current memory bank, obtaining an adapted user profile $\mathcal{M}_q$. A separate frozen LLM (π_{gen}) then produces the final response $\hat{a} = \pi_{gen}(q, \mathcal{M}_q)$.

To autonomously execute the three memory decisions defined above, SALIMORY trains a single, unified policy model π_θ via RL rather than relying on separate specialized models. By conditioning π_θ on distinct instruction prompts, this single model learns to act as the saliency filter, the memory booster, and the memory utilizer.

We formally define the *state* at any given step as the combination of the conversational context and the memory bank $(q, \mathcal{H}, \mathcal{M})$. The *action space* corresponds to the generative outputs of our policy model π_θ , including the saliency decision, the updated memory $\mathcal{M}$, and the query-dependent profile $\mathcal{M}_q$. Our goal for the RL training is to manipulate the state through these actions to maximize the quality of the final downstream response $\hat{a}$.

3.3 STAGE-WISE PROCESS REWARD SYSTEM

Recent advances in RL for reasoning models (Guo et al., 2025) rely heavily on Reinforcement Learning with Variable Reward (RLVR), where the model is optimized purely on a final, outcome-based reward. However, applying pure RLVR to a complex, multi-stage memory pipeline creates a severe credit assignment bottleneck. If the frozen generator produces an incorrect or poorly personalized final answer, an outcome-only reward cannot determine where the pipeline failed.

To solve this, SALIMORY decomposes the learning signals into a dense, hierarchical stage-wise process reward, providing targeted, orthogonal feedback to each specific role using LLM judges.

Reward 1: Response Quality (Multiplicative Gating) The primary objective evaluates the final generated answer $\hat{a}$ against the ground truth a for accuracy $\alpha \in \{0, 1\}$ and personalization quality $p \in [0, 1] \cup \{\texttt{n/a}\}$. We formulate this reward using a strict multiplicative gate:

$$R_1 = \begin{cases} \alpha \cdot (1 + \lambda_p \cdot p) & \text{if personalization is applicable} \\ \alpha & \text{otherwise} \end{cases} \tag{1}$$

This multiplicative form embeds a critical inductive bias: personalization is only valuable when the underlying answer is factually correct. By multiplying the terms, an incorrect but highly personalized response receives $R_1 = 0$, preventing the policy from learning to "decorate" hallucinated answers.

Reward 2: Memory Quality (Asymmetric Penalty) To directly supervise the memory booster's writing behavior independently of the final answer, a judge evaluates the finalized memory bank $\mathcal{M}_K$ for factuality ϕ_f (fraction of entries faithful to the source) and vagueness ϕ_v (fraction of entries losing actionable specificity). Defining the composite memory quality as $\gamma = \phi_f \cdot (1 - \phi_v)$, we apply

an asymmetric reward:

$$R_2 = -\lambda_{\text{pen}} \cdot (1-\gamma) + \lambda_{\text{bon}} \cdot \gamma, \qquad \text{where } \lambda_{\text{pen}} > \lambda_{\text{bon}} \tag{2}$$

The asymmetry $\lambda_{\text{pen}} = 2 \times \lambda_{\text{bon}}$ reflects the fact that low-quality memories actively sabotage downstream generation. This asymmetric penalty exerts a conservative pressure on the booster: it learns that it is strictly better to omit a marginal memory than to encode a vague one.

Reward 3: Utilization and Filter Quality The final level provides targeted, isolated feedback to the utilizer and the saliency filter. It is defined as:

$$R_3 = \lambda_\mu \cdot \mu + \lambda_\kappa \cdot \kappa \tag{3}$$

Here, $\mu \in [0,1]$ evaluates how effectively the adapted profile $\mathcal{M}_q$ captures question-relevant constraints. Meanwhile, κ evaluates the saliency filter's binary gating decisions against ground-truth labels using a recall-weighted blend: $\kappa = (1-w_r) \cdot \text{Prec} + w_r \cdot \text{Rec}$ because missing a salient turn (false negative) is worse than flagging an extra non-salient one (false positive).

Total Reward The total per-episode reward $R_{total} = R_1 + R_2 + R_3$. The key property is *role-specific credit assignment*: R_1 (range $[0, 1+\lambda_p]$) dominates as the primary objective, R_2 (range $[-\lambda_{\text{pen}}, \lambda_{\text{bon}}]$) directly shapes the booster's writing behavior independently of response quality, and R_3 (range $[0, \lambda_\mu + \lambda_\kappa]$) separately shapes the utilizer's synthesis and the filter's gating.

3.4 Reward-Decomposed Contrastive Learning

Standard GRPO assigns a single advantage scalar to all generative traces within a rollout—affecting the filter, booster, and utilizer equally even though we use role-specific process rewards. Consequently, a rollout might achieve a high total reward simply due to a fortunate generation by the utilizer, which inadvertently sends the upstream booster an unearned positive gradient even if it writes vague or poor memories. To fully resolve this intra-episode credit assignment bottleneck, SaliMory exploits our reward decomposition to build *role-specific contrastive preference pairs*, attributing credit to each role from its own contribution.

Booster Contrastive Pairs ($\mathcal{L}_{\text{cont}}^{\text{boost}}$) For the memory booster, we select winner (w) and loser (l) rollouts based strictly on the R_2 memory quality score, completely ignoring the total episode reward. Let $\mathcal{P}_b$ be the set of paired booster traces: at each turn where the two rollouts produce different memory writes, the winner's output y_j^w is paired with the loser's y_j^l under the same booster prompt p_j^b. We apply a DPO-style contrastive loss over these pairs, scaled by a gap weight $\Delta_{R_2} = |R_2^w - R_2^l|$:

$$\mathcal{L}_{\text{cont}}^{\text{boost}} = \frac{1}{|\mathcal{P}_b|} \sum_{j \in \mathcal{P}_b} \Delta_{R_2} \cdot \Big[-\log \sigma\Big(\beta\big(\log \pi_\theta(y_j^w \mid p_j^b) - \log \pi_\theta(y_j^l \mid p_j^b)\big)\Big)\Big] \tag{4}$$

The gap weight modulates the gradient magnitude by preference confidence: when the quality gap between two memory writes is large, the contrastive signal strongly pushes the booster; when marginal, the loss contributes minimally, avoiding noisy updates from ambiguous comparisons.

Utilizer Contrastive Pairs & Similarity Gating ($\mathcal{L}_{\text{cont}}^{\text{util}}$) For the utilizer, winner and loser rollouts are selected based on the utilization score μ (from R_3). However, a naive comparison here is causally confounded: a superior adapted profile might simply be the result of the booster providing better upstream memories to work with. To isolate the utilizer's true synthesis ability from upstream memory noise, we introduce a gating mechanism based on the Jaccard similarity between the memory banks of the two rollouts:

$$\text{sim}(\mathcal{M}^w, \mathcal{M}^l) = \frac{|\text{entries}(\mathcal{M}^w) \cap \text{entries}(\mathcal{M}^l)|}{|\text{entries}(\mathcal{M}^w) \cup \text{entries}(\mathcal{M}^l)|} > \tau_{\text{sim}} \tag{5}$$

Contrastive pairs are only formed if this similarity exceeds the threshold τ_{sim}. This strict similarity gate ensures that any difference in profile quality reflects the utilizer's own synthesis ability—how it selected, prioritized, and phrased the context—rather than differences in the underlying available information. Below-threshold pairs are discarded, so the utilizer only receives gradients it actually earned. With $\mathcal{P}_u$ denoting the admitted profile pairs and $\Delta_\mu = |\mu^w - \mu^l|$, the utilizer contrastive loss is:

$$\mathcal{L}_{\text{cont}}^{\text{util}} = \frac{1}{|\mathcal{P}_u|} \sum_{(w,l) \in \mathcal{P}_u} \Delta_\mu \cdot \Big[-\log \sigma\Big(\beta\big(\log \pi_\theta(\rho^w \mid p^u) - \log \pi_\theta(\rho^l \mid p^u)\big)\Big)\Big] \tag{6}$$

where ρ is the adapted profile and p^u the utilizer prompt.

These two role-specific contrastive refinements combine to form our additional learning boost:

$$\mathcal{L}_{\text{Cont}} = \lambda_b \cdot \mathcal{L}_{\text{cont}}^{\text{boost}} \; + \; \lambda_u \cdot \mathcal{L}_{\text{cont}}^{\text{util}} \tag{7}$$

Ultimately, this boost does not replace the main RL objective, but acts alongside it to provide the precise credit assignment necessary to train a highly structured cognitive pipeline.

3.5 Full Objective and Policy Optimization

The complete per-step loss seamlessly integrates our stage-wise process rewards into the standard GRPO (Shao et al., 2024) objective with our novel contrastive boost:

$$\mathcal{L}_{\text{total}} = \frac{1}{N} \sum_{\xi} \mathcal{L}_{\text{GRPO}}(\xi) \; + \; \mathcal{L}_{\text{Cont}} \tag{8}$$

where N is the total number of generative traces and $\mathcal{L}_{\text{GRPO}}$ is the standard clipped surrogate objective with KL regularization optimized using our hierarchical rewards R_{total}.

Crucially, this combined objective finalizes our solution to the credit assignment problem through selective gradient routing. The main GRPO loss $\mathcal{L}_{\text{GRPO}}$ provides the baseline update and trains the saliency filter, whereas $\mathcal{L}_{\text{Cont}}$ backpropagates only through the booster- and utilizer-conditioned traces of the shared policy, based on their isolated preference pairs.

4 Experimental Setup

Dataset. To evaluate SaliMory, we need a dataset with Chat AI scenarios that provide conversation history, current query and ground truth response. Thus, we used LoCoMo (Maharana et al., 2024), a long-term conversational memory benchmark with 10 users and ∼58 multi-session dialogues per user. The original benchmark provides query types across five categories (multi-hop, single-hop, open-domain, temporal, adversarial) of *memory recall* queries. We argue that personalized engagements are crucial for conversational AI, yet they are absent from existing memory datasets such as LoCoMo and LongMemEval (Wu et al., 2024). To address this, we extend LoCoMo and introduce a new dataset named **LoCoMo-P13n** with two personalization categories—*recommendation* and *implicit personalization*, curated to support personalization research (Appendix A.1 provides more details). Additionally, to validate the generalizability of SaliMory, we evaluated on an internal dataset of real-world Chat AI traffic, as described in Section 5.4.

Models and training. The trainable policy π_θ uses Qwen3.5-9B-Instruct (Team, 2026); the frozen generator π_{gen} uses Qwen3-235B-A22B-Instruct (Yang et al., 2025) and the judge uses GPT-4o (Hurst et al., 2024). For fair comparison, we reproduce all baselines with the same settings; for zero-shot agentic method, we use Gemini-3-flash (Pichai et al., 2025) to play as the agent for memory management and utilization. Appendix A.6 gives full training details and hyperparameter setup [1].

Baselines and ablations. To validate the superiority of SaliMory compared to SOTA memory studies, we compared it against four existing distinct popular memory paradigms: **Infinite Context**, which treats the entire history as memory; **RAG** – A-Mem (Xu et al., 2025b), MemoryGAS (Xu et al., 2025a), which process history into a memory base for future use; **Zero-shot Agentic**, which uses powerful LLM as agent for memory operation; and **Learnable Memory** – Mem-R1 (Yan et al., 2025), which learns to memorize and utilize user info in an RL framework.

Evaluation Protocol. To comprehensively assess memory-driven conversational agents, we introduce a novel LLM-as-a-Judge protocol that thoroughly evaluates fine-grained personalization and memory bottlenecks, with full implementation details provided in Appendices A.3 and A.5.

- End-to-End Personalization Quality: Standard QA accuracy (Jiang et al., 2025) fails to capture user experience. Thus, we introduce a *Three-Step Personalization Judge*. Step 1 determines if the response incorporates user memories. Step 2 classifies personalized responses as *Good* (rich and value-adding), *Basic* (superficial), or *Bad* (intrusive, irrelevant, or hallucinated). Step 3 evaluates unpersonalized responses to determine if relevant memories exist but were ignored (*Underuse*), or there is genuinely no personalization potential (*NA*).

[1] Backbone models and training ablation analysis are in Appendix C; Judge quality validation is in A.4

Table 1: **Main results on LoCoMo and LongMemEval.** Best in **bold**. ↑/↓: higher/lower is better.

	Response Quality						Memory Quality	
	QA	Personalization Categories (%)						
Method	Accuracy ↑	Good ↑	Basic	Bad ↓	Underuse ↓	NA ↓	Vague ↓	Factuality ↑
(a) Comparative Results on ***LoCoMo and LoCoMo-P13n***								
Infinite Context	33.4	18.2	13.6	**0.8**	39.3	27.3	-	-
Zero-shot Agentic	70.9	6.6	20.9	6.1	42.8	23.6	29.7	86.1
A-Mem	61.6	21.9	**49.1**	6.4	8.5	13.1	44.2	44.3
MemoryGAS	49.1	5.1	8.2	11.6	8.9	66.2	46.7	59.3
Mem1	15.2	1.6	7.9	19.6	17.1	53.8	59.3	31.6
Mem-α	46.1	16.0	31.4	6.4	9.5	36.7	36.2	72.6
Mem-R1	62.7	16.3	21.8	7.4	19.9	34.6	35.7	76.8
SALIMORY	**72.9**	**39.8**	44.1	1.4	**5.3**	**9.4**	**19.6**	**93.9**
(b) Comparative Results on ***LongMemEval-m***, results on ***LongMemEval-s*** are in Appendix C								
Infinite Context	17.3	4.6	19.1	1.3	42.6	32.4	-	-
Zero-shot Agentic	62.2	13.6	22.2	2.8	11.2	50.2	9.6	79.6
A-Mem	49.4	12.2	21.2	2.0	11.4	53.2	27.7	45.6
MemoryGAS	51.2	7.3	19.4	2.1	15.2	56.0	21.9	57.3
Mem1	50.4	8.0	38.8	1.8	13.8	37.6	11.6	56.0
Mem-α	57.6	14.6	34.4	2.4	15.8	32.8	9.5	80.2
Mem-R1	61.5	14.1	37.6	3.9	27.1	17.3	9.9	81.9
SALIMORY	**66.4**	**22.6**	**39.3**	**1.3**	**6.2**	**30.1**	**9.3**	**89.7**

- Intermediate Memory Eval: We evaluate the finalized memory bank for *Vagueness* (entries lacking actionable specificity) and *Factuality* (faithfulness to the source). We also measure *Utilization*, scoring how well the synthesized profile captures question-relevant constraints.
- Gated Error-Bucket: For every incorrect response, we trace the failure to its origin—Memory Error, Profile Error, or Generation Error—to pinpoint specific reasoning bottlenecks.

Combined together, to the best of our knowledge, SALIMORY is the first framework to thoroughly evaluate fine-grained personalization and memory bottlenecks in conversational agents.

5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we answer the following three questions. More results can be found in Appendix C:

- **RQ 1.** Does SALIMORY exhibit better performance over state-of-the-art solutions?
- **RQ 2.** What contributes to the improvements made by SALIMORY?
- **RQ 3.** How does SALIMORY generalize to different types of queries in Chat AI scenarios?

5.1 (RQ1) COMPARATIVE STUDY

End-to-End Quality. As presented in Table 1, SALIMORY achieves the highest E2E Accuracy (72.9%) on memory recall questions and highest Good Personalization rate (39.8%) on personalizable questions on LoCoMo benchmarks. Though trained on a small backbone model (Qwen3.5-9B), SALIMORY outperforms the Zero-shot baseline powered by a much larger model Gemini-3-Flash—delivering a 33% gain in Good Personalization. If we consider Good personalization and Basic personalization together, SALIMORY also outperforms all other methods, including A-Mem, where most personalization is only *Basic* but not *Good* yet. Furthermore, SALIMORY nearly eradicates the Bad Personalization rate (1.4% vs. Mem-R1's 7.4%), proving our R1 reward prevents the model from misusing personal information. On LongMemEval dataset, the findings are consistent, SALIMORY again attains the highest QA accuracy (66.4%) and Good personalization rate (22.6%).

Prior memory works fail for different reasons. Infinite Context provides all information but fails at utilization (39.3% underuse) because raw logs overwhelm the reasoning capacity. Conversely, methods like MemGAS and Mem-R1 focus heavily on memory formation but struggle to effectively retrieve and apply it, resulting in high NA rates (66.2% and 34.6%). *Why does* SALIMORY *help?*

Table 2: **RL training ablation studies.** Best in each section in **bold**. ↑/↓: higher/lower is better.

	QA	Personalization Quality			Memory Quality		Memory
Config	Accuracy ↑	Good↑	Basic	Bad↓	Vague↓	Factuality↑	**Utility ↑**
(a) Reward composition on SALIMORY							
R1 only	65.8	30.6	41.7	4.5	28.3	85.4	0.747
R1 + R2	69.2	35.7	41.9	3.6	22.4	91.7	0.799
R1 + R2 + R3	**72.9**	**39.8**	44.1	**1.4**	**19.6**	**93.9**	**0.917**
(b) Contrastive refinement							
GRPO only	68.6	26.8	39.5	3.2	31.7	90.9	0.793
+ Booster contr. only	69.1	29.5	41.7	3.2	20.6	92.4	0.799
+ Utilizer contr. only	71.7	35.9	41.7	1.8	22.1	91.7	0.906
+ Both (full SALIMORY)	**72.9**	**39.8**	44.1	**1.4**	**19.6**	**93.9**	**0.917**

By orchestrating memory into cognitively-structured stores and applying a role-decomposed RL framework, SALIMORY explicitly co-optimizes both memory formation and cue-driven utilization, drastically reducing NA (9.4%) and underused (5.3%) responses.

Memory Quality. SALIMORY has a low Vague memory rate (19.6%, vs. 35.7% for Mem-R1) and a high Factual rate (93.9%). This validates that SALIMORY successfully filters noise during memory formation. While the Gemini-3-Flash agentic baseline achieves decent memory factuality (86.1%) and high QA Accuracy (70.9%), its dismal personalization score shows that without dedicated training for memory utilization, even massive frontier models can under-utilize memory context. Following our stage-wise process reward and contrastive refinement, SALIMORY bridges this gap, enabling a significantly smaller open-source model to effectively couple memory formation with downstream utilization. The memory-quality advantage generalizes across distributions: SALIMORY records the lowest Vague rate (9.3%) and highest Factuality (89.7%) on LongMemEval, confirming that the asymmetric R2 penalty consistently suppresses noisy memories.

5.2 (RQ2) ABLATION STUDY

To understand what contributes to the improvements made by SALIMORY, we conducted extensive ablation studies on reward compositions, contrastive refinement components, and memory types.

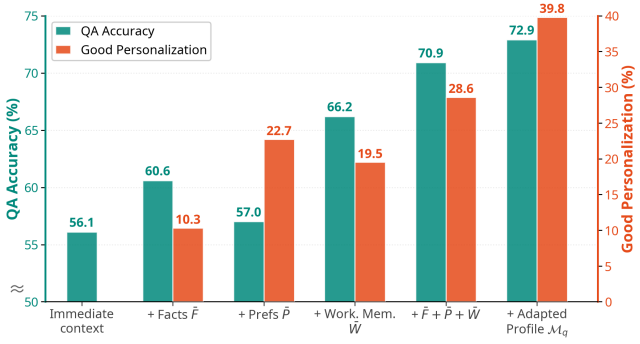

Figure 3: Impact of memory types on end performance

Reward composition. Table 2(a) validates our stage-wise process reward. Using only R1 (end-to-end) yields Accuracy of 65.8% and Utilization of 0.747. Adding R2 (memory quality) improves Accuracy to 69.2% and drops Vague from 28.3% to 22.4%. Incorporating R3 (utilization) brings the system to full performance (72.9%, 0.917). This confirms that *sparse global signals are inadequate for training intermediate memory operations*—explicit stage-wise supervision is necessary. **Contrastive refinement components.** As shown in Table 2(b), Booster contrastive training specifically improves memory formation (raising Factuality to 92.4%) while dropping the Vague rate from 31.7% to 20.6%. Conversely, Utilizer contrastive training specifically enhances retrieval (boosting Good Personalization to 35.9% and utilization score to 0.906). Combining both ensures each module only receives gradients it earned, yielding full system performance.

Memory type comparison. As illustrated in Figure 3, isolating the memory banks reveals their distinct cognitive roles. The Working Memory bank primarily drives QA Accuracy (66.2%), while the Preference bank drives Good Personalization (22.7%). Crucially, merely combining these raw

banks bottlenecks Good Personalization at 28.6%. It is only when the Utilizer synthesizes these stores into a query-adapted profile that personalization leaps to 39.8%, proving that active context construction is strictly superior to raw retrieval.

5.3 Gated Error Analysis

We conduct a gated error analysis to isolate where failures occur in the pipeline (Figure 1). Memory creation errors account for 37.9% of failures in MemGAS and 25.3% in Mem-R1, validating that poor memory formation is the primary bottleneck in existing systems. SaliMory not only provides the lowest memory (16.4%) and profile generation (5.8%) error rates, but it pushes the overall system to the highest correctness rate (72.9%). *Why does* SaliMory *succeed?* By enforcing selective attention and using the asymmetric R2 penalty, SaliMory actively prevents vague or hallucinated information from polluting the memory stores. The remaining failures are largely isolated to the frozen generator after curing the upstream memory bottleneck, confirming our memory management module effectively fulfills its role.

5.4 (RQ3) Efficiency & Generalizability Analysis

Efficiency Analysis. Beyond quality, practical deployment demands strict efficiency. As shown in Table 3(a), the Zero-shot baseline indiscriminately stores 127.4 generic memories per user, creating a severe computational bottleneck with 38 seconds of memory formation latency. In contrast, SaliMory's saliency filter and booster actively condense context into just 89.7 highly specific entries. The deliberate saliency filtering reduces memory formation latency to 0.5 seconds—a 76× speedup. At runtime, inference latency drops nearly 2× (0.69s to 0.26s), proving that feeding the generator a synthesized profile is vastly more efficient than processing raw retrieval chunks.

Table 3: **(a)** Efficiency and memory store statistics; **(b)** Results on real-world Chat AI traffic data.

Metric	**Zero-shot**	**SaliMory**
(a) Efficiency Analysis		
Avg. memories per user	127.4	89.7
Avg. memory token length	8.6	10.9
Avg. adapted memories per query	1.9	1.7
Fact : Pref : WM ratio	21:11:68	9:14:77
Memory formation latency (s/episode)	38	0.5
Training latency (s/episode)	-	1.14
Inference latency (s/query)	0.69	0.26
(b) Generalizability Analysis		
Good P13n%	1.7	5.5
Vague%	63.9	46.1
Factuality%	41.9	60.7

Real-World Generalizability. We deployed SaliMory on an internal dataset of real-world Chat AI traffic as in A.1. As shown in Table 3(b), the relative margins remain highly consistent. Even in this noisy, unstructured setting, SaliMory more than triples the Good Personalization rate (1.7% → 5.5%), drops the Vague memory rate from 63.9% to 46.1%, and spikes memory Factuality from 41.9% to 60.7%, proving SaliMory produces a high-quality memory which leads to better end performance even compared to the powerful agent today. This confirms that SaliMory's cognitive architecture and role-decomposed training generalize robustly beyond benchmark-specific distributions.

6 Conclusion

We presented SaliMory, an agentic learnable memory-management framework that addresses the fundamental challenge of conversational memory: not what to store, but how to manage it. Our work introduces three key novelties. First, a **cognitively-inspired memory architecture** that separates factual snapshots, subjective preferences, and working memory, enabling structured reasoning over heterogeneous user information. Second, a **stage-wise, role-decomposed RL** training paradigm that combines hierarchical process rewards with reward-decomposed contrastive refinement, directly resolving the credit-assignment problem that has limited prior RL-based memory agents. Third, an **extended evaluation protocol** and a new benchmark **LoCoMo-P13n** are proposed to jointly stress precise recall and nuanced personalization across diverse query types in Chat AI. Empirically, SaliMory outperforms strong baselines—while operating at 2x lower inference latency than the zero-shot baseline. These results demonstrate that memory management, grounded in cognitive science and trained with decomposed rewards, is the key to build lifelong conversational agents.

7 STATEMENTS

AI USE STATEMENT

In paper writing, we used LLMs solely for checking typos and grammar. In our experiments, an LLM (Llama-3.3-70B-Instruct) helped synthesize personalizable queries (e.g., "Can you recommend a place to visit this coming weekend?"). We also used AI to assist with debugging during experiments. No other use of AI was made beyond the above.

ETHICS STATEMENT

After carefully reviewing the ICLR Ethics Policy, we are committed to showing our respect and obtaining consents from all parties.

REPRODUCIBILITY STATEMENT

We release our implementations of SALIMORY, the curated LoCoMo-P13n dataset, and all baselines at `https://anonymous.4open.science/r/SaliMory-4D59` to promote reproducibility and support the open-source community. Detailed training and data curation procedures are provided in Appendices A.1 and A.6.

REFERENCES

Richard C Atkinson and Richard M Shiffrin. Human memory: A proposed system and its control processes. *Psychology of Learning and Motivation*, 2:89–195, 1968.

Alan Baddeley. The episodic buffer: A new component of working memory? *Trends in Cognitive Sciences*, 4(11):417–423, 2000.

Alan D. Baddeley and Graham Hitch. Working memory. In Gordon H. Bower (ed.), *The Psychology of Learning and Motivation: Advances in Research and Theory*, volume 8, pp. 47–89. Academic Press, New York, 1974.

Sanghwan Bae, Donghyun Kwak, Soyoung Kang, Myoung-Wan Lee, Sungdong Kim, Yun Jeong, Hyungjoo Kim, Eunho Lee, and Jungwoo Seo. Keep me updated! memory management in long-term conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 3769–3781, 2022.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pp. 2206–2240. PMLR, 2022.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Prateek Chhikara, Aditya Khant, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Hongda Jiang, Xinyuan Zhang, Siddhant Garg, Rishab Arora, Shiun-Zu Kuo, Jiayang Xu, Aaron Colak, and Xin Luna Dong. Memory-qa: Answering recall questions based on multimodal memories. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 24255–24277, 2025.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25972–25981, 2025.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. *arXiv preprint arXiv:2402.09727*, 2024.

Hao Li, Chenghao Yang, An Zhang, Yang Deng, Xiang Wang, and Tat-Seng Chua. Hello again! llm-powered personalized agent for long-term dialogue. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5259–5276, 2025.

Jiying Li et al. LD-Agent: Longitudinal dialogue agent with self-evolving memory. *arXiv preprint arXiv:2406.18484*, 2024.

John Lisman, Anthony A Grace, and Dheeraj K Bhatt. A neohebbian framework for episodic memory; role of dopamine-dependent late LTP. *Trends in Neurosciences*, 34(10):536–547, 2011.

Adyasha Maharana, Dong-Ho Lee, Sergey Tuber, Amir Rahimi, Mohit Joshi, Khyathi Raghavi Chandu, and Yunyao Dong. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pp. 13064–13080, 2024.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. Memgpt: towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Sundar Pichai, Demis Hassabis, and Koray Kavukcuoglu. A new era of intelligence with gemini 3. *Google. URL:https://blog.google/products-and-platforms/products/gemini/gemini-3/)*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Daphna Shohamy and R Alison Adcock. Dopamine and adaptive memory. *Trends in Cognitive Sciences*, 14(10):464–472, 2010.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Endel Tulving. Episodic and semantic memory. *Organization of Memory*, pp. 381–403, 1972.

Yu Wang, Xinshuang Liu, Xiusi Chen, Sean O'Brien, Junda Wu, and Julian McAuley. Self-updatable large language models by integrating context into model parameters. *arXiv preprint arXiv:2410.00487*, 2024.

Yuxin Wang, Ryuichi Takanobu, and Minlie Huang. Mem-α: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, Tong Xu, et al. Towards multi-granularity memory association and selection for long-term conversational agents. *arXiv preprint arXiv:2505.19549*, 2025a.

Junfeng Xu, Yuxiang Liang, and Qiaozhu Mei. A-Mem: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*, 2025b.

Xinchao Xu, Zhibin Gou, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, and Shihang Wang. Long time no see! open-domain conversation with long-term persona memory. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2639–2650, 2022.

Ming Yan, Yiming Yang, and Qipeng Huang. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

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.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. Agent lumos: Unified and modular training for open-source language agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12380–12403, 2024.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025.

Kai Zhang, Xinyuan Zhang, Ejaz Ahmed, Hongda Jiang, Caleb Kumar, Kai Sun, Zhaojiang Lin, Sanat Sharma, Shereen Oraby, Aaron Colak, et al. Assomem: Scalable memory qa with multi-signal associative retrieval. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026b.

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.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. *arXiv preprint arXiv:2506.15841*, 2025.

A Experimental Setup Details

A.1 Dataset Details

Table 4 summarizes the extended LoCoMo dataset. The original benchmark (Maharana et al., 2024) provides QA pairs across four evaluation categories (multi-hop, temporal, open-domain, single-hop); we contribute two additional categories—recommendation and implicit personalization—that require memory-driven personalization, adding 1,108 QA pairs (42% of the total) to form a new dataset named **LoCoMo-P13n**. Category 5 (adversarial/unanswerable) is excluded following prior work. Each user's conversation history spans multiple sessions with natural temporal gaps.

Table 4: Dataset statistics.

Statistic	**Value**
Users	28
Total QA pairs	2,637
Original (Cat 1–4)	1,529
P13n extension (Cat 6–7)	1,108
Train / eval / test	60 / 20 / 20%
Max turns per episode	80
Per-category counts:	
Multi-hop	282 (10.7%)
Temporal	321 (12.2%)
Open-domain	96 (3.6%)
Single-hop	830 (31.5%)
Recommendation†	603 (22.9%)
Implicit P13n†	505 (19.2%)

†Our contributed annotations.

How we construct the data : we use LLM, LlaMa-3.2-70B, to semi-automatically generate the data following a 4-step strategy:(1) extracting relevant information from conversations, (2) identifying conversation reference points, (3) aggregating related information, and (4) composing questions that—rather than directly querying conversation content—utilize extracted signals to enable personalized or recommended responses. We have a linguistic expert with background in data science to do manually review on **LoCoMo-P13n**.

Algorithm 1 SALIMORY: Training Step

Require: Batch of episodes $\{(\mathcal{H}_e, q_e, a_e^*)\}$, policy π_θ, reference π_{ref}, generators/judges $\pi_{\text{gen}}, \pi_{\text{judge}}$

for each episode e, each rollout $g = 1, \ldots, G$ **do**
  Phase A (Filter): $s_t \leftarrow \pi_\theta^{\text{filt}}(u_t)$ for all t; $\mathcal{S} \leftarrow \{t : s_t = \text{True}\}$
  Phase B (Booster): **for** $k = 1, \ldots, |\mathcal{S}|$: $\langle d_k, v_k, o_k \rangle \leftarrow \pi_\theta^{\text{boost}}(u_{t_k}, \mathcal{M}_{k-1})$; $\mathcal{M}_k \leftarrow \text{Apply}(\mathcal{M}_{k-1}, d_k, v_k, o_k)$ ▷ t_k: *k-th turn in* $\mathcal{S}$
  Phase C (Utilizer): $\mathcal{M}_q \leftarrow \pi_\theta^{\text{util}}(\mathcal{M}_K, q_e)$ ▷ $K = |\mathcal{S}|$
  Cache $\log \pi_\theta^{\text{old}}(y \mid P)$ for all traces; pre-compute $\log \pi_{\text{ref}}(y \mid P)$
end for
Rewards: $\hat{a} \leftarrow \pi_{\text{gen}}(q_e, \mathcal{M}_q)$; compute R_1, R_2, μ via π_{judge}, κ from ground-truth labels ▷ *API, concurrent*
Advantages: $\tilde{A}_{e,g} \leftarrow (R_{e,g} - \bar{R}_e)/\sigma_{R_e}$ for each episode e, where $R_{e,g} = R_1 + R_2 + R_3$ ▷ *GRPO group norm.*
Contrastive pairs: Construct $\mathcal{P}_b$ by R_2 (weighted by Δ_{R_2}); construct $\mathcal{P}_u$ by μ (weighted by Δ_μ), gated on $\text{sim}(\mathcal{M}^w, \mathcal{M}^l) > \tau_{\text{sim}}$
for $i = 1, \ldots, E$ **do**
  Compute $\mathcal{L}$ via Eq. 8; update θ ▷ *inner epoch*
  if mean KL $> 1.5\times$ target **then**
    break ▷ *KL early stopping*
  end if
end for

Real-world Chat AI dataset To verify the generalizability of SALIMORY, we further collect data from real world Chat AI traffic where real users are producing conversations. We use their past 28-days chat history as the user raw history to form memory. The dataset comprises 2,719 examples spanning 69,444 sessions and 110,996 turns in total, with a median of 22 sessions (range 2–83) and 36 turns (range 5–100) per example. Sessions are short, containing 1.6 turns on average, with each turn averaging 9.8 words. Each example is additionally paired with a question-answer pair (median question/answer length of 11/31 words) and an immediate conversation context (median 8 turns, mean 18.0). Unfortunately, this dataset is not able to be open-sourced due to privacy constraint, but the results in Section 5.4 verify the effectiveness of SALIMORY in real-world application.

A.2 SALIMORY TRAINING ALGORITHM

SALIMORY training steps are shown in Algorithm 1. To ensure the complex multi-stage pipeline does not bottleneck compute, our training step utilizes GPU and API resources complementarily. First, the policy generates traces entirely on the GPU, and reference log-probabilities are pre-computed and cached. Second, the frozen generator and reward judges run concurrently via API, allowing the reference model to be safely offloaded to the CPU. Finally, during the update phase, only the policy model requires GPU memory since all reference probabilities were previously cached. Each training step performs E inner epochs over this cached rollout data, executing early stopping if the mean KL divergence exceeds $1.5\times$ the target threshold.

A.3 THREE-STEP PERSONALIZATION JUDGE

Standard single-score LLM-as-judge evaluation conflates whether a response *uses* personal information with whether it uses it *well*, and cannot detect cases where relevant memories existed but went unused. We design a three-step sequential judge that disentangles these dimensions (4).

Task 1: Personalization Detection. A binary classifier determines whether the response incorporates personal information from the user's memory beyond what is already present in the query or immediate conversation. The judge considers both explicit usage (e.g., "considering you like hiking, I recommend...") and implicit usage (e.g., recommending a casual restaurant when the user's memory includes a preference for casual dining). Crucially, information that is already present in the user's query or conversational context does not count as personalization—the response must draw on stored memories. If personalized, proceed to Task 2; otherwise, proceed to Task 3.

Task 2: Personalization Quality (if personalized). For personalized responses, a multi-criteria judge evaluates five dimensions: (1) *richness*—does the personalization go beyond simply inserting user information, using multiple memory signals in a natural way? (2) *value-add*—does the personalization enhance the response? (3) *style*—is the tone appropriate? (4) *relevance*—is the response on-topic? (5) *quality*—is the response coherent and non-hallucinated? These combine into three quality levels:

- **Good**: rich personalization (criterion 1) with value-add (criterion 2) and good overall response quality (criteria 3–5).
- **Basic**: personalization is present but not rich or not value-adding; overall response quality is acceptable. Core case: using the user's name without deeper integration.
- **Bad**: personalization is misapplied (intrusive, irrelevant, or offensive) or the overall response quality is poor despite personalization.

Task 3: Signal Detection (if not personalized). For unpersonalized responses, the judge determines whether relevant memory signals *existed* that could have enhanced the response but were not used. This detects *underuse*—a failure mode invisible to standard evaluation. If no relevant signals exist (e.g., a purely factual query or no applicable user memory to personalize the query), the response is simply marked as not applicable (NA).

Five-category taxonomy. The three tasks yield five mutually exclusive response categories, shown in 4.

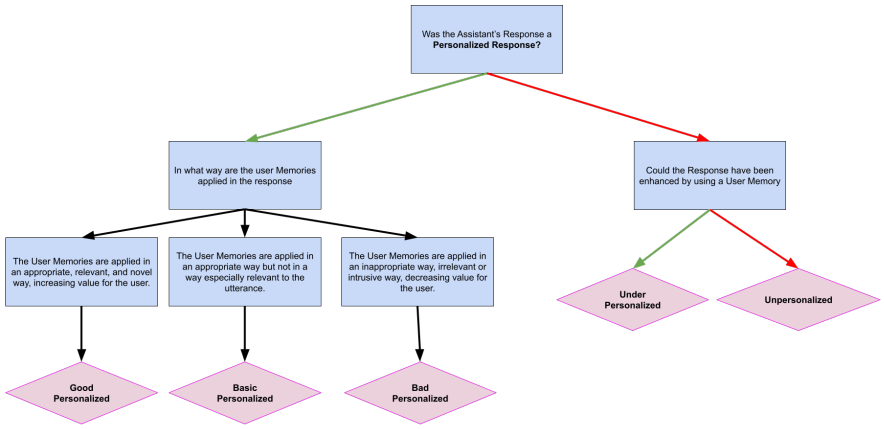

Figure 4: **Three-step personalization judge flowchart.** Task 1 detects whether the response uses memory-based personalization. If yes, Task 2 evaluates quality (Good / Basic / Bad). If no, Task 3 checks whether relevant signals existed but went unused (Underuse) or were genuinely absent (Unpersonalized). The five leaf categories are mutually exclusive.

We derive three end-to-end personalization metrics from this taxonomy: **P13n Rate** (fraction of responses in any personalized category, including bad), **Good P13n Rate** (fraction in GOOD PERSONALIZED—the strictest measure), and **Underuse Rate** (fraction in UNDERUSE—measuring missed opportunities).

Human Agreement Test To validate the trustworthiness of this proposed judge, two linguistic engineers from our team conducted human agreement tests on the LoCoMo-P13n and our internal datasets. We have seen an accuracy score of 91%, 89% and a kappa score of 0.71, 0.69 on the two datasets, respectively. The results indicate the evaluation protocol is substantially trustful for our experiments.

A.4 JUDGE CROSS-VALIDATION

To verify that our E2E findings are not an artifact of GPT-4o judge bias, we rerun the full E2E evaluation (QA accuracy and the three-step personalization judge) on a random subset of 200 test QA pairs from LoCoMo/LoCoMo-P13n using Claude 3.5 Sonnet as an independent judge. As shown in Table 5, absolute scores shift only slightly between judges (within 2.5%), with Claude marginally stricter on QA accuracy, but the method ranking is identical under both judges (Spearman's $\rho = 1.00$ for both QA accuracy and Good personalization rate), and SALIMORY remains the top method on both metrics under both judges. This confirms that the reported gains reflect genuine differences in memory quality and utilization rather than judge-specific preferences.

Table 5: E2E results on a random 200-query subset of the LoCoMo/LoCoMo-P13n test set under two independent judges. Best in **bold**.

Method	GPT-4o		Claude 3.5 Sonnet	
	QA ↑	Good ↑	QA ↑	Good ↑
SALIMORY	**73.4**	**40.5**	**71.8**	**38.2**
Zero-shot Agentic	70.2	7.1	69.5	6.3
Mem-R1	63.1	15.8	62.4	16.9
A-Mem	60.8	22.4	60.1	20.7
MemoryGAS	48.5	5.6	49.8	4.9
Mem-α	46.9	15.2	45.3	16.1
Infinite Context	34.1	17.5	32.8	18.9
Mem1	14.6	2.0	16.1	1.4

A.5 METRIC DEFINITIONS

Table 6 provides definitions for all reported metrics.

Table 6: Evaluation metrics. E2E personalization metrics are derived from the three-step judge (A.3); memory metrics are judged independently.

Metric	**Level**	**Definition**
QA Accuracy	E2E	Fraction of responses judged correct against ground truth
P13n Rate	E2E	Fraction in any personalized category (A.3)
Good P13n Rate	E2E	Fraction in GOOD PERSONALIZED (A.3)
Underuse Rate	E2E	Fraction in UNDERUSE (A.3)
Vague Rate	Memory	Fraction of memory entries too generic to be actionable
Factuality Rate	Memory	Fraction of entries faithful to source utterances
Profile Util. Score	Memory	Judge-scored relevance of profile ρ to question q ($[0, 1]$)
Error Bucket	Diagnostic	Gated attribution of each failure to the earliest failing stage: memory → profile → generation

Memory quality scoring. The judge evaluates each entry in $\mathcal{M}_K$ against the original utterance that produced it. An entry is *vague* if it has lost the specificity needed to be actionable (e.g., "User likes food" vs. "User loves Thai curry"). An entry is *non-factual* if it misrepresents the source (e.g., "thinking about getting a dog" → "owns a dog").

Error-bucket attribution. For each incorrect response, we apply a gated cascade: if the profile contains sufficient information to answer correctly → GENERATION ERROR; else if the memory contains the information but the profile missed it → PROFILE ERROR; else → MEMORY ERROR. Each failure is assigned to the earliest failing stage.

A.6 HYPERPARAMETERS

For training, we use transformers and PyTorch to train on a computation node with 8 A100 GPUs.

Table 7: Hyperparameters for SALIMORY training.

Group	Parameter	Value
Training	GRPO group size G	4
	Batch size	8
	Training epochs	5
	Inner update epochs E	4
Optimization	Learning rate	10^{-5}
	Optimizer	AdamW
	Warmup ratio	0.1
	Gradient clip norm	1.0
RL	Clip ratio ϵ	0.2
	KL coefficient λ_{KL}	0.08
	KL target (early stopping)	1.5
Reward	Personalization weight λ_p	0.45
	Memory penalty λ_{pen}	0.5
	Memory bonus λ_{bon}	0.3
	Utilization weight λ_μ	0.15
	Filter weight λ_κ (recall w_r=0.6)	0.1
Contrastive	Booster weight λ_b	0.15
	Utilizer weight λ_u	0.1
	Temperature η	0.15
	Memory similarity threshold τ_{sim}	0.15

B LIMITATIONS

Despite the success brought by SALIMORY, we still notice two limitations to be further studied in the future: **Dependence on a frozen large LLM for judging.** All three reward levels rely on a frozen large LLM (GPT-4o) — response accuracy, memory quality, and utilization are all assessed via LLM-as-a-judge calls. This means the training signal is bounded by the judge model's own evaluation capability and biases and further, the api calling would hinder the training stability. **Sequential memory formation limits its scalability to long conversations.** The booster must process salient turns sequentially (each turn needs the updated memory state from the previous one), meaning memory formation latency scales linearly with conversation length. For the current dataset (16 turns/episode), this is manageable, but for real-world deployment with hundreds or thousands of turns per user, this becomes a bottleneck that batch parallelization cannot resolve.

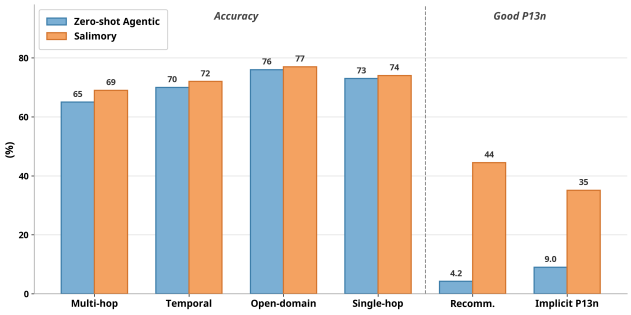

Figure 5: **Per-category accuracy.** SALIMORY improves all categories, with the largest gains on recommendation and multi-hop questions that require memory synthesis.

C SUPPLEMENTARY ANALYSIS

C.1 COMPARATIVE RESULTS ON LONGMEMEVAL-S

C.2 BACKBONE MODEL AND TRAINING STRATEGY ABLATION

Backbone models. Table 9(b) shows that Llama3.2-1B collapses (Accuracy 29.6%, Vague 74.1%), Llama3.2-3B achieves 58.8% Accuracy, and Qwen3.5-9B reaches the optimal 72.9%. This suggests

Table 8: **Main results on LongMemEval-s.** Response quality and memory quality are reported. Best in **bold**. ↑/↓: higher/lower is better.

	Response Quality						Memory Quality	
	QA	Personalization Categories (%)						
Method	Acc. ↑	Good ↑	Basic	Bad ↓	Under. ↓	NA ↓	Vague ↓	Fact. ↑
(b) Comparative Results on ***LongMemEval-s***								
Infinite Context	47.7	9.6	22.8	5.7	42.9	19.0	-	-
Zero-shot Agentic	70.9	8.0	27.3	6.8	40.6	17.3	25.5	89.1
A-Mem	60.0	13.6	22.9	1.7	12.5	49.3	21.1	53.7
MemoryGAS	60.2	7.6	18.1	2.0	19.2	53.1	19.9	58.1
Mem1	51.4	14.6	31.6	2.9	18.0	32.9	15.9	81.6
Mem-α	54.8	13.8	32.2	2.2	19.0	32.8	9.6	82.2
Mem-R1	66.7	15.1	37.2	4.5	29.3	13.9	9.9	82.8
SALIMORY	**69.1**	**20.8**	**43.7**	**0.8**	**6.8**	**27.9**	**6.3**	**93.3**

Table 9: SALIMORY results using different training strategies and backbone models. Best in each section in **bold**. ↑/↓: higher/lower is better.

	E2E Quality			Memory Quality		Memory Utilization
Config	Acc.↑	Basic P13n	Good P13n↑	Vague↓	Fact.↑	Util.↑
(a) Training strategy						
REINFORCE	36.9	23.7	12.9	68.6	43.9	0.291
PPO	39.2	23.9	15.7	67.9	48.6	0.393
GRPO	68.6	39.5	26.8	31.7	90.9	0.793
SALIMORY	**72.9**	44.1	**39.8**	**19.6**	**93.9**	**0.917**
(b) Backbone models						
Llama3.2-1B	29.6	9.7	0.9	74.1	40.3	0.148
Llama3.2-3B	58.8	19.4	11.7	39.1	71.5	0.476
Qwen3.5-9B	**72.9**	44.1	**39.8**	**19.6**	**93.9**	**0.917**

that the cognitive memory roles require a minimum threshold of base reasoning capacity to execute effectively. We probe the base policy with 200 canary prefixes from LoCoMo test conversations; verbatim continuations occur in 0% of cases, indicating no measurable memorization.

Training strategy. As presented in Table 2(a), REINFORCE and PPO yield Accuracy of only 36.9% and 39.2%, with Vague rates remaining high at 68.6% and 67.9%. Switching to GRPO provides a massive leap (Accuracy 68.6%, Vague 31.7%), indicating that relative scoring within a batch is crucial for stable learning in long-horizon memory tasks. *Why does* SALIMORY *push further?* By augmenting GRPO with contrastive refinement, SALIMORY achieves 72.9% Accuracy and 19.6% Vague rate, verifying that standard GRPO alone is insufficient to fully resolve credit assignment in multi-stage memory management.

C.3 FROZEN-GENERATOR ABLATION

Our main results use a frozen Qwen3-235B-A22B-Instruct generator, raising the question of whether the gains stem from generator capacity rather than the learned memory module. We rerun E2E evaluation with the trained SALIMORY policy held fixed and the generator swapped to Qwen3.5-9B-Instruct, matching the policy's scale. As shown in Table 10, absolute scores drop for all methods under the weaker generator, as expected, but SALIMORY retains the lead on both QA accuracy and Good personalization, with its QA margin over Mem-R1 widening from 10.2pp to 12.5pp. Memory-quality metrics (Vague, Factuality) are unaffected by construction, since they are judged on the policy-produced memory bank. This confirms that the improvements originate in the memory-

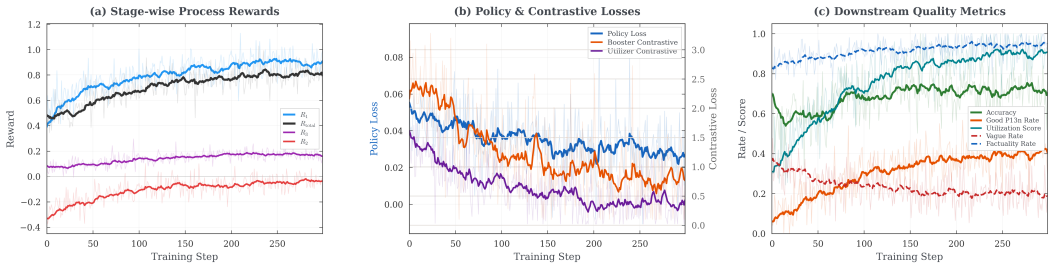

Figure 6: **Training dynamics of SALIMORY.** (a) Stage-wise process rewards. (b) Policy and contrastive losses. (c) Downstream quality metrics. Curves are EMA-smoothed.

management module rather than in generator scale—high-quality, well-utilized memories matter more, not less, when the generator is weaker.

Table 10: E2E results with the default large generator versus the policy-matched small generator on LoCoMo/LoCoMo-P13n. Best in **bold**.

Method	Qwen3-235B-A22B		Qwen3.5-9B	
	QA ↑	Good ↑	QA ↑	Good ↑
SALIMORY	**72.9**	**39.8**	**64.8**	**34.2**
Zero-shot Agentic	70.9	6.6	58.6	4.8
Mem-R1	62.7	16.3	52.3	12.1
A-Mem	61.6	21.9	50.2	17.3

C.4 MEMORY COMPARISON ANALYSIS

We directly validate that memory quality drives downstream performance by comparing retrieval under controlled memory conditions (11). Filtering vague entries improves Recall@5 from 35.9% to 52.1% and good-personalization from 22.2% to 40.9%. Vague memories act as retrieval noise, displacing useful entries from top-k results. SALIMORY's booster directly addresses this via the asymmetric R_2 penalty.

C.5 TRAINING DYNAMICS ANALYSIS

Figure 6 illustrates SALIMORY's training dynamics. Panel (a) reveals a bottom-up causal learning chain: the memory quality reward (R_2) improves first (rising from -0.26 to -0.04) before the utilization reward (R_3) and the response reward (R_1) follow. Panel (b) shows both contrastive losses converging (booster: 2.6→0.7, utilizer: 1.4→0.35), demonstrating that role-decomposed preference signals provide targeted credit assignment. Panel (c) validates that these training signals translate to downstream quality: Vague Rate drops to 19.6%, Utilization Score reaches 0.917, and Good P13n Rate improves 6× to 39.8%, with the improvement order mirroring the reward progression.

Table 11: **Impact of vague memories.** Filtering vague entries nearly doubles good-personalization rate.

Memory Condition	Recall@5↑	Win Rate↑	P13n↑	Good P13n↑
With vague memories	35.9	34.6	29.6	22.2
Without vague memories	**52.1**	**40.9**	**49.3**	**40.9**

Table 12: **Infinite context degrades performance.** More memory increases personalization *attempts* but collapses their quality.

Context	Win Rate↑	P13n↑	Good P13n↑	Mis-rec↓
Top-5 retrieval	40.9	49.3	40.9	8.3
Infinite context	27.6	52.6	22.7	29.6

C.6 More Memory ≠ Better Responses

Table 12 tests whether providing all memories (infinite context) yields an upper bound on performance. Win rate drops from 40.9% to 27.6% and mis-recommendation rate spikes from 8.3% to 29.6%. The generator attempts *more* personalization (P13n rate rises) but gets it wrong (good P13n collapses). This motivates the utilizer's role: compress and filter memories into a question-relevant profile rather than passing raw context.

C.7 Per-Category Breakdown

Figure 5 reports performance by question category. SaliMory achieves the largest accuracy gains on multi-hop (+4pp) and recommendation (+5pp)—categories requiring multi-entry synthesis—while single-hop and open-domain show minimal improvement (+1pp), as single-fact retrieval already works well. For personalization, SaliMory lifts Good P13n from 4.2% to 44.5% on recommendation and from 9.0% to 35.1% on implicit personalization, confirming that our cognitive memory architecture and trained utilizer generalize most strongly to synthesis-heavy queries.

D Reward Judge Prompts

All reward signals in SaliMory are computed by frozen LLM judges. Below we provide the judge prompt templates for each reward level.

D.1 R1 Response Quality Judge

R1 Response Quality

Context You are a judge who evaluates a chatbot response on two independent axes: accuracy and personalization.
The user has the following personal information: Memory: memory
Here is the immediately preceding conversation: conversation
Here is the user query: question
Here is the response to evaluate: response
Here is the golden response (ground truth): label
Evaluation Criteria
1. Correctness (boolean) Does the response answer the query correctly? Compare against the golden response. - If the response is factually wrong or fails to answer the question, correctness is false. - If the response addresses the question correctly (even if worded differently from the golden response), correctness is true.
2. Accuracy Score (0.0–1.0) How correct is the response compared to the golden answer? This is independent of personalization. - 0.0: Completely wrong or fails to answer the question. - 0.3: Partially addresses the question but with significant errors. - 0.5: Partially correct with some relevant information. - 0.7: Mostly correct with minor gaps. - 1.0: Fully correct, equivalent to the golden response.
3. Personalization Score (0.0–1.0 or "n/a") How well does the response use personal information from memory? - 0.0: Misused or irrelevant personal information that hurts the response. - 0.3: Personal info acknowledged but not meaningfully integrated. - 0.5: Superficial use of personal information. - 0.7: Good use of relevant personal information that enhances the response. - 1.0: Excellent use of personal information, comparable to the golden response. - "n/a": The question does not require personalization, or no relevant memory exists.
Output Provide your evaluation as a JSON object with: - 'correctness': Boolean (true/false). - 'accuracy_score': String, a number from 0.0 to 1.0. - 'personalization_score': String, a number from 0.0 to 1.0, or "n/a" if not applicable. Output the JSON object only, nothing else.

D.2 R2 MEMORY QUALITY JUDGE

R2 Memory Quality

Task Description You are a professional memory analyst. Your job is to evaluate a memory collection extracted from a conversation on two axes: vagueness and factuality.
Key Definitions
Vagueness - **Vague memory entry**: A memory entry that is too generic to be actionable for personalization. It loses specific details from the original conversation. For example, "User likes food" is vague because it lost what kind of food; "User traveled" is vague because it lost where and when. - **Specific memory entry**: A memory entry that captures concrete, actionable information that can personalize future responses. For example, "User has a peanut allergy" or "User returned from Japan trip last week" are specific.
Factuality - **Factual memory entry**: A memory entry that accurately represents what was said in the conversation, without hallucinating or distorting details. - **Non-factual memory entry**: A memory entry that misrepresents what was said, invents details, or draws incorrect conclusions.
Vagueness Examples
Conversation utterances: - Can you recommend some Thai restaurants near downtown? - I just got back from a trip to Japan last week - I'm training for a marathon in October
Vague memory: "user_facts": ["User likes food", "User is interested in running"], "working_memory": ["User traveled recently"] -¿ 3 entries, all 3 are vague -¿ vague_rate = 1.0
Specific memory: "user_facts": ["User has a peanut allergy", "User is training for a marathon in October"], "working_memory": ["User returned from Japan trip last week", "User is looking for Thai restaurants near downtown"] -¿ 4 entries, 0 are vague -¿ vague_rate = 0.0
Factuality Examples
Utterance: "I went to Japan last week" -¿ Memory: "User went to China last week" -¿ **Non-factual** (wrong country) Utterance: "I'm thinking about getting a dog" -¿ Memory: "User owns a dog" -¿ **Non-factual** (considering is not owning) Utterance: "My daughter starts college in September" -¿ Memory: "User's daughter is starting college in September" -¿ **Factual**
Input Conversation utterances: utterances
Extracted memory: memory
Instructions 1. List all individual memory entries across all categories (user_facts, user_preference, working_memory, etc.). 2. For each entry, determine if it is vague or specific. Calculate: vague_rate = (number of vague entries) / (total number of entries). If there are no entries, vague_rate = 0.0. 3. For each entry, determine if it is factual or non-factual based on the conversation. Calculate: factuality_rate = (number of factual entries) / (total number of entries). If there are no entries, factuality_rate = 1.0.
Output Output a JSON object with: - 'vague_rate': A string representing a number from 0.0 to 1.0, indicating the fraction of memory entries that are vague. - 'factuality_rate': A string representing a number from 0.0 to 1.0, indicating the fraction of memory entries that are factual. Output the JSON object only, nothing else.

D.3 R3 UTILIZATION QUALITY JUDGE

R3 Utilization Quality

Task Description You are a professional analyst evaluating the quality of a user profile summary for answering a specific question.
Context A utilizer agent has created a user profile summary from memory and conversation to help answer a question. Your job is to evaluate how useful this profile is for generating a correct, personalized answer.
Input Question: question
User Profile Summary: user_profile
Memory State: memory_state
Conversation: conversation
Golden Answer (ground truth): ground_truth
Evaluation Rubric (0.0–1.0) - 0.0: Profile is irrelevant or contains wrong information that would mislead the answer. - 0.3: Profile exists but misses key information needed to answer the question. - 0.5: Contains some relevant information but incomplete or noisy. - 0.7: Captures the essential information needed for answering the question correctly. - 1.0: Highly relevant, accurate, and provides exactly the right context for a personalized answer.
Output Output a JSON object with: - 'utilization_score': A string representing a number from 0.0 to 1.0. Output the JSON object only, nothing else.