AMEMGYM: INTERACTIVE MEMORY BENCHMARKING FOR ASSISTANTS IN LONG-HORIZON CONVERSATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon interactions between users and LLM-based assistants necessitates effective memory management, yet current approaches face challenges in training and evaluation of memory. Existing memory benchmarks rely on static, off-policy data as context, limiting evaluation reliability and scalability. To address these gaps, we introduce AMEMGYM, an interactive environment enabling on-policy evaluation and optimization for memory-driven personalization. AMEMGYM employs structured data sampling to predefine user profiles, state-dependent questions, and state evolution trajectories, enabling cost-effective generation of high-quality, evaluation-aligned interactions. LLM-simulated users expose latent states through role-play while maintaining structured state consistency. Comprehensive metrics based on structured data guide both assessment and optimization of assistants. Extensive experiments reveal performance gaps in existing memory systems (e.g., RAG, long-context LLMs, and agentic memory) and corresponding reasons. AMEMGYM not only enables effective selection among competing approaches but also can potentially drive the self-evolution of memory management strategies. By bridging structured state evolution with free-form interactions, our framework provides a scalable, diagnostically rich environment for advancing memory capabilities in conversational agents.

1 INTRODUCTION

A crucial objective in the development of assistants based on Large Language Models (LLMs) is to achieve long-horizon conversational capabilities—that is, the ability to effectively organize, manage, and utilize memory across extended sequences of dialogue turns. Robust memory management forms the foundation for fulfilling complex user requests, tailoring responses to users' latest implicit states, and personalizing suggestions and recommendations based on interaction history. However, progress in advancing conversational memory systems for assistants is hampered by a critical bottleneck that affects both scalable training and reliable evaluation: the data used in existing benchmarks.

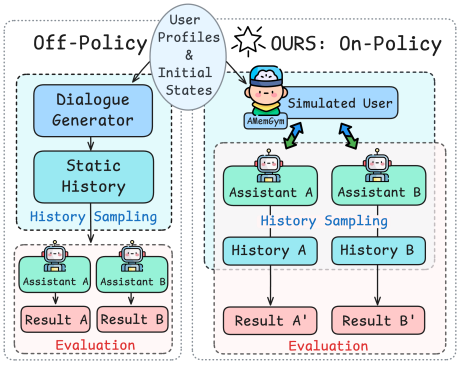


Figure 1: On-policy v.s. off-policy evaluation for assistants' memory.

Current benchmarks typically rely on static, off-policy data for evaluation (Xu et al., 2022; Wu et al., 2024; Hu et al., 2025), rather than on-policy interactions. Figure 1 shows the comparison on two approaches. Off-policy evaluation, in which an assistant is tested on conversational data that it did not produce during actual interactions, presents several fundamental limitations. First, it fails to capture the assistant's true interactive property, as the evaluation data does not reflect the consequences of the as-

Table 1: A comparison of features across agent memory benchmarks.

Benchmark	Eval. Mode	Optim. Feedback	Automation Level	Context Length	Eval. Metrics
MSC (Xu et al., 2022)	Static	✗	Manual	1.2K	-
RealTalk (Lee et al., 2025)	Static	✗	Manual	17K	Emotional Intelligence, Persona Simulation, Memory Probing (F1, accuracy)
DialSim (Kim et al., 2024)	Static	✗	Manual	-	QA Accuracy
LoCoMo (Maharana et al., 2024)	Static	✗	Semi-Automated	9.2K	QA Accuracy, Summarization, Generation
PerLTQA (Du et al., 2024)	Static	✗	Semi-Automated	-	QA Accuracy
LongMemEval (Wu et al., 2024)	Static	✗	Semi-Automated	Configurable (115K, 1.5M)	Retrieval Recall, QA Accuracy
PersonaMem (Jiang et al., 2025)	Static	✗	Fully Automated	Configurable (32K, 128K, 1M)	QA Accuracy
AMEMGYM (This Work)	Interactive	✓	Fully Automated	Configurable	Overall (Accuracy, Normalized Memory Score) and Diagnosis (Write, Read, Utilization).

sistant's own conversational choices—a critical issue for evaluation realism. Second, because the evaluation is biased, memory optimization could be misguided to wrong directions. Finally, the manual curation of these evaluation scenarios (Lee et al., 2025; Kim et al., 2024) is costly and does not scale for comprehensive testing across diverse, long-horizon conversational contexts.

To enable on-policy evaluation and provide reliable feedback for optimization, it is essential to employ a simulated user that can strategically reveal information and pose relevant questions, a technique that has demonstrated promise in other domains such as tool use (Wang et al., 2023; Lu et al., 2025). However, deploying simulated users in open-ended conversational environments presents unique challenges. These include determining what information to disclose dynamically while maintaining a natural and coherent dialogue, as well as ensuring the generation of diverse, high-quality data that remains sufficiently controlled for reliable evaluation.

To address these gaps, we introduce AMEMGYM, an interactive environment designed for the on-policy evaluation and optimization of memory in long-horizon conversations. AMEMGYM grounds free-form interactions in structured data generated through a schema-based approach. The framework predefines user profiles, state-dependent questions, and state evolution trajectories to enable the cost-effective generation of high-quality interactions aligned with evaluation targets. LLM-simulated users then expose these latent states through natural role-play, ensuring consistency with the structured state evolution. Periodic evaluation during interactions, using both overall and diagnostic metrics, guides assessment and optimization of memory capabilities. Our contributions are threefold:

1. We introduce AMEMGYM, a novel framework for the on-policy evaluation of conversational memory. By grounding free-form interactions in a structured state evolution, AMEMGYM creates a scalable and diagnostically rich environment to reliably assess and advance the memory capabilities of conversational agents.
2. We empirically demonstrate the reuse bias and potential drawbacks of off-policy evaluation, and conduct the first **extensive on-policy evaluation** of popular memory systems. Our results highlight the reliability of AMEMGYM for evaluating memory in the context of personalization.
3. We provide a proof of concept for **agent self-evolution**, showing that an agent can use environmental feedback within AMEMGYM to autonomously refine its memory management policy.

2 RELATED WORK

Benchmarks for agent memory evaluation. The evaluation of agent memory has progressed from long-context, single-turn tasks like the needle-in-a-haystack (NIAH) test and NoLiMa (Modarressi et al., 2025) to more realistic multi-turn conversational datasets such as Multi-Session Chat (MSC) (Xu et al., 2022), RealTalk (Lee et al., 2025), and DialSim (Kim et al., 2024). While these introduced more authentic dialogue patterns, their reliance on manual curation limited their scale and diversity. To address this, automated data generation frameworks like LoCoMo (Maharana et al., 2024), PerLTQA (Du et al., 2024),

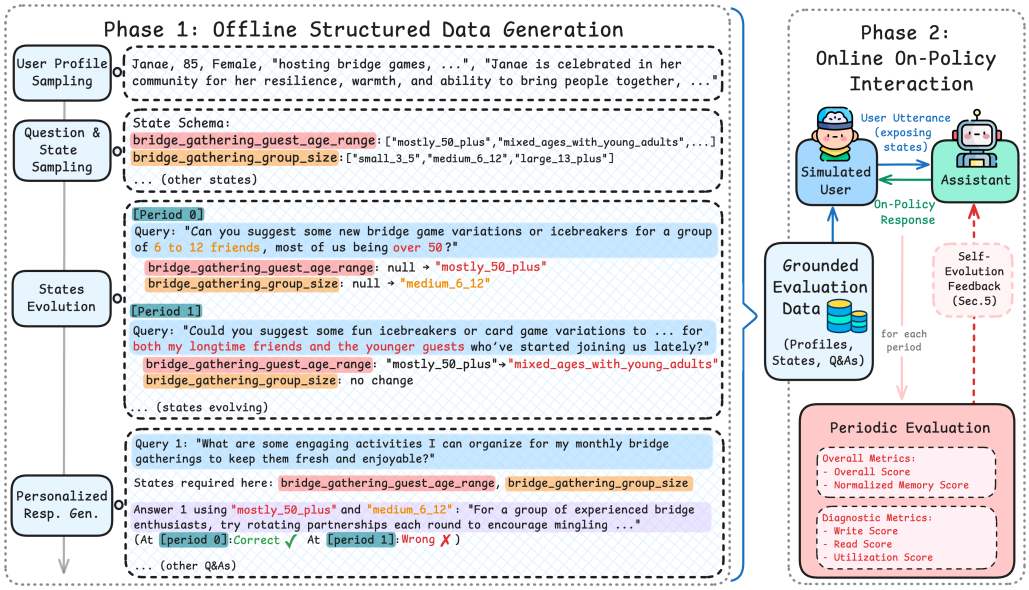


Figure 2: An overview of the AMemGym framework.

LongMemEval (Wu et al., 2024), PersonaMem (Jiang et al., 2025), and MemoryAgentBench (Hu et al., 2025) were developed. However, a critical limitation unites nearly all existing benchmarks: they rely on static, off-policy data (Table 1). This approach fails to capture an agent's true interactive performance, as the evaluation data does not reflect the consequences of the agent's own actions, misleading optimization.

Interactive agent evaluation by user simulation. An alternative line of research has focused on interactive, on-policy evaluation environments that employ user simulators. This approach has proven effective in domains like tool-use, where simulators provide robust on-policy evaluation (Wang et al., 2023; Lu et al., 2025). Similarly, efforts like CollabLLM (Wu et al., 2025) have successfully employed user simulation to train models for improved long-term collaboration, enabling them to move beyond passive responses and actively help users achieve their goals. Applying this interactive paradigm to memory evaluation, however, introduces unique challenges: a simulator must strategically reveal information over a long-horizon conversation while maintaining a natural flow and generating interactions that are both diverse and controlled enough for reliable assessment. AMemGym directly addresses these challenges by introducing a schema-based approach that grounds free-form, LLM-driven role-play in a structured state evolution plan, which enables the controlled and scalable generation of on-policy, memory-focused evaluation scenarios.

3 AMemGym

AMemGym provides an interactive environment for benchmarking and optimizing personal assistant memory, with the scenario and the task described below.

LLM-based Assistants. An LLM-based assistant takes as input the observation (user input) o_t and provides output responses a_t (a sequence of tokens) based on its policy π and its internal memory at that time m_t (e.g., tokens in the context window, text snippets written to an external index, or its own parameters): $o_t, m_t \xrightarrow{\pi} a_t, m_{t+1}$. The internal memory is updated through interactions.

Personalization with Memory. To effectively serve users with dynamically evolving personal states, assistants described above must continuously track user states through interaction histories $\tau_t = [o_0, a_0, o_1, a_1, \ldots, o_t]$ and deliver responses optimized for their latest latent states captured by m_t. In reality,

the length of τ_t often goes well beyond the optimal context length of most LLMs. Therefore, an effective information compression or memory mechanism is crucial for assistants to maintain accurate and up-to-date user modeling. Here, *states* refer to comprehensive personal information crucial for enabling the intelligent assistant to sustain meaningful conversations and address user-relevant concerns. This includes user preferences, habits, plans, and environmental conditions, among other factors.

An overview of our framework[1] is presented in Figure 2. We begin by describing the structured data sampling process that forms the foundation of our evaluation framework (§ 3.1), then detail how on-policy interactions are generated with grounded structured data (§ 3.2). We present comprehensive evaluation metrics that assess both overall memory performance and provide diagnosis for different memory operations (§ 3.3). Finally, we provide meta-evaluation results to show reliability of the fully-automated process (§ 3.4).

3.1 Generating Structured Data for On-Policy Interaction

Evaluating memory is challenging due to the high cost of verifying correctness in long, noisy conversations. To address this, we use a reverse-engineering strategy: starting from target evaluation questions, we trace back to identify key user state variables for personalization, their possible temporal changes for a simulated user, and the personalized responses for each experienced state combination. This servers as a structured foundation that enables grounded interactions and automatic evaluation. Detailed prompts for each sampling step are provided in Appendix C.1.

User Profile Sampling. We begin by selecting user profiles, which provide background information for subsequent steps. For broad domain coverage, we use 100K personas from Nemotron-Personas (Meyer & Corneil, 2025) as the pool, but custom sampling strategies can be easily applied for specific applications to better accommodate target real-world distributions.

Question Sampling. The process starts with a user profile, p, used to sample a set of evaluation questions, $\mathcal{Q}_p$. For each question $q_i \in \mathcal{Q}_p$, an LLM extracts the information types required for a personalized answer. These types $\mathcal{S}'_i$ are occasionally redundant across questions (e.g., "experience_level" and "years_of_work"). Therefore, they are merged and refined by an LLM into a canonical **global state schema**, $\Sigma = \bigcup_i \mathcal{S}'_i$. The schema defines a set of M unique state variables (s_j) and their possible discrete values set (V_j): $\Sigma = \{(s_j, V_j)\}_{j=1}^{M}$. This comprehensive schema serves as the complete set of trackable user states for the entire simulation.

User States Evolution. We then simulate a realistic progression of the user's states over N_p periods. The state at the end of each period t is captured by a **state vector**, σ_t, a full assignment where each variable s_j is given a value v_j from its corresponding set of possibilities V_j: $\sigma_t = \{(s_j, v_j) \mid (s_j, V_j) \in \Sigma\}$. Each state transition is prompted by a narrative **life event**, e_t, providing context for the change ($\sigma_{t-1} \xrightarrow{e_t} \sigma_t$). The resulting **state evolution trajectory**, $\mathcal{T}_\sigma = (\sigma_0, \ldots, \sigma_{N_p})$, provides the ground-truth for the user's state throughout the simulation.

To create the inputs for on-policy interaction in each session, we generate a series of natural language utterances that the simulated user will say initially. Within each period t, an utterance $u_{t,k}$ is designed to implicitly *expose* a small related subset of the user's current state, $\sigma_{\text{exposed}} \subset \sigma_t$. This is generated by a function G_{utt} conditioned on the states to be revealed and the user's profile: $u_{t,k} = G_{\text{utt}}(\sigma_{\text{exposed}}, p)$. These pre-generated, state-bearing utterances form a core part of the structured data blueprint. They are used to initiate conversational turns during the on-policy interaction phase (§ 3.2).

Personalized Response Generation. Finally, to create the evaluation ground truth, we generate personalized answers for each predefined question q_i. Each question requires a subset of state variables, $\mathcal{S}_{\text{req}}(q_i) \subset \{s_1, \ldots, s_M\}$, and a specific assignment of values to these variables is a **state variant**, ν:

[1] We use gpt-4.1 (OpenAI, 2025) for structured data generation and user simulation.

$\nu = \{(s_j, v_j) \mid s_j \in \mathcal{S}_{\text{req}}(q_i), v_j \in V_j\}$. For each pair (q_i, ν), we generate a distinct answer $r_{i,\nu}$. To ensure a high-quality, one-to-one mapping, a reflection step verifies that the answer is unambiguous: it is accepted only if an LLM classifier C can recover the variant from the question-answer pair, i.e., $C(q_i, r_{i,\nu}) = \nu$.

3.2 On-policy Interaction

Different from prior static evaluation on long-context LLMs or memory agents (Xu et al., 2022; Maharana et al., 2024; Wu et al., 2024; Jiang et al., 2025), we sample on-policy interactions as in Figure 1. Given the offline structured data sampled in Section 3.1, our user simulator interacts with the target assistant to expose this information through natural conversation. This step outputs a (possibly long-context) dialogue history τ. Later in Section 4.2, we demonstrate the necessity of on-policy evaluation.

State Exposure. To enable reliable evaluation, key user states—those that change between periods—must be clearly reflected in the conversation history. This is achieved by using the grounded utterances $(u_{t,k})$ that were pre-generated as part of the structured data. For benchmarking consistency, we use these fixed initial state-bearing utterances to begin each conversational session, ensuring that the necessary information is introduced into the dialogue.

Role-Play with LLMs. Conversation generation is performed by a user LLM, which role-plays based on the user profile and state evolution. It is configured with: (1) a system prompt template incorporating the user profile, (2) current states σ_t, and (3) the latest conversation context. The user LLM produces responses conditioned on dialogue history and underlying states, ensuring coherent alignment between free-form conversation and structured state evolution.

3.3 Evaluation Metrics

Given the grounded interactive environment, assistants are prompted to answer all evaluation questions after each interaction period. These responses provide feedback for agent builders to assess and optimize assistants (§ 4), and enable assistants to self-improve (§ 5), based on the evaluation metrics described below.

Overall Evaluation. We use the average question answering accuracy as the metric for evaluating end-to-end performance on our benchmark, denoted as the *overall* score. This metric captures the model's ability to integrate both personalization (tailoring responses based on specific user states) and memory (retaining user states from previous conversations) to achieve high performance. To provide a clearer view on memory, we introduce normalized *memory* scores. It isolates the memory component from raw task performance by normalizing the overall accuracy between a random baseline (lower bound) and an upper bound (UB) with perfect memory access. For each evaluation period, the score is computed as: $S_{\text{memory}} = \frac{S_{\text{overall}} - S_{\text{random}}}{S_{\text{UB}} - S_{\text{random}}}$. The upper bound S_{UB} is determined by providing the assistant with ground-truth user states at evaluation time, thereby entirely bypassing the memory retrieval process. It measures the assistant's reasoning and application capabilities when required information is perfectly available.

Diagnostic Evaluation. We decompose failures in overall question answering into three distinct operational stages of memory processing: *write*, *read*, and *utilization*. Corresponding failure rates enable systematic error attribution. For each user state, we query its value at every evaluation period. If the assistant demonstrates knowledge of all relevant state values but still fails to answer an overall evaluation question correctly, we classify this as a *utilization* failure. Otherwise, we examine the state query results at the nearest write position to distinguish between *write* and *read* failures (Figure 3).

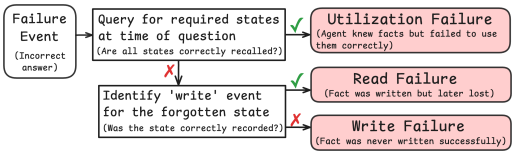


Figure 3: An overview of diagnostic metrics: *write, read,* and *utilization*.

3.4 META-EVALUATION

To validate the data quality of AMEMGYM, we conducted a two-stage meta-evaluation with human annotators. First, we assessed *state exposure*, confirming that user states are clearly introduced into the conversation. On a sample of 200 queries, annotators found that the state information was successfully conveyed with an average quality score of 99.1% and an inter-annotator agreement (Gwet's AC1 (Gwet, 2001)) of 96.8%. Second, we evaluated *conversational state integrity* to ensure that the simulated user's dialogue does not contradict established ground-truth states over time. Across 748 annotated items from 40 conversations, the dialogue maintained a 99.2% consistency score, with a Gwet's AC1 of 98.2%. These results confirm that AMEMGYM generates high-fidelity data, providing a reliable foundation for memory evaluation. Details of this evaluation are in Appendix D.

4 MEMORY EVALUATION WITH AMEMGYM

4.1 EVALUATION SETUP

Data Configuration. AMEMGYM offers configurable parameters to control evaluation difficulty. We focus on two configurations to showcase flexibility and ensure reproducibility, differing in three key dimensions: the number of evolution periods N_p (quantity of key information), required states per question N_s (reasoning depth), and interaction turns per state exposure N_i (noise level). We define two variants using the tuple (N_p, N_s, N_i): *base* (10, 2, 4) which requires 128K+ context window and *extra* (20, 3, 10) which requires 512K+ context window. Both variants use 20 randomly sampled user profiles with 10 evaluation questions each, totaling 200 questions tested at $N_p + 1$ positions with potentially different answers due to evolving user states. We present *base* results in the main text as they are sufficiently challenging. See Appendix F.1 for *extra* results and other configurable parameters.

Memory Implementation. Existing memory systems for LLM-based assistants, despite implementation variations, share a common design philosophy of constructing memory hierarchies to exchange between short-term and long-term memory (Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025). We abstract this connection by focusing on two key aspects: storage location (in-context vs. external) and writing strategy (agentic vs. direct).

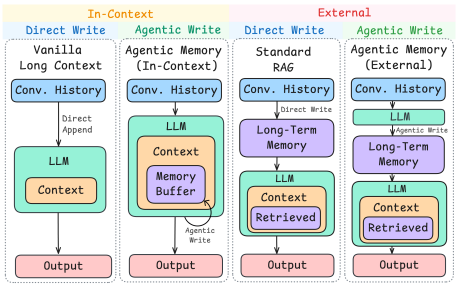


Figure 4: Memory implementations.

As shown in Figure 4, we focus on the four memory implementations: *Native LLMs* (**LLM**) rely solely on context windows, maintaining long-term memory in-context as raw content. *Standard RAG* (**RAG**) uses Retrieval-Augmented Generation with external indexing for long-term storage in raw format. *Agentic Write (External)* (**AWE**) autonomously decides what to write to external long-term memory and retrieves using embedding models as in RAG. *Agentic Write (In-Context)* (**AWI**) operates similarly but stores long-term memory in-context without independent retrieval. For **AWE**, we additionally study critical parameters: memory update frequency (*freq*), minimum short-term messages in-context (*ns*), and retrieved memories count (*topk*).[2] We denote these configurations as AWE-(freq, ns, topk).[3] All memory implementations use gpt-4.1-mini (OpenAI, 2025) for response generation and memory operations and text-embedding-3-small (OpenAI, 2024a) for embeddings to ensure a fair comparison.

[2]We implement *AW* and *RAG* variants using the open-source mem0 library (Chhikara et al., 2025).

[3]We use AWE-(2,4,30) as the default configuration.

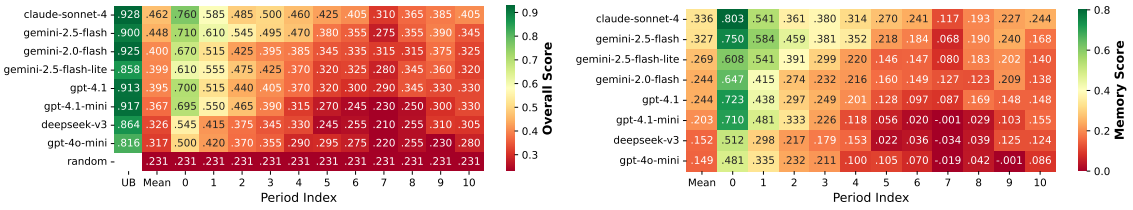


Figure 5: Evaluation on native LLMs. Overall scores and normalized memory scores are both demonstrated.

We evaluate a diverse set of native LLMs, including claude-sonnet-4 (Anthropic, 2025), gemini-{2.5-flash, 2.5-flash-lite, 2.0-flash} (Google, 2024; 2025), gpt-{4.1, 4.1-mini} (OpenAI, 2025), deepseek-v3 (Liu et al., 2024), and gpt-4o-mini (OpenAI, 2024b). All models are configured with max tokens as 8192 and temperature as 0. The prompts used for evaluation are provided in Appendix C.3. For user simulation, we employ gpt-4.1 and the additional study presented in Appendix F.2 indicate that the choice of user LLM has minimal impact on the evaluation results.

4.2 On-policy versus Off-policy Evaluation

Off-policy evaluation introduces reuse bias, undermining memory optimization and configuration selection, particularly for agents. All existing memory benchmarking studies use off-policy evaluation, testing models on pre-generated interaction traces that do not reflect their own conversational behavior. We directly compare on-policy and off-policy evaluation with AMemGym, where off-policy evaluation uses on-policy interaction traces from gpt-4.1 for memory updates and omits the interaction process.

Table 2: The on-policy v.s. off-policy comparison on memory scores of various assistants. Results on different native LLMs are listed in a separate table below. Memory agents use the same LLM (gpt-4.1-mini) for generation.

Memory Agents	On-policy ↑	Off-policy ↑	ΔRank
AWE-(2,4,30)	.291	.253(.038)	▼ 3
AWE-(2,8,30)	.278	.271(.007)	–
AWE-(2,4,10)	.275	.273(.002)	▲ 2
AWE-(4,4,30)	.262	.229(.033)	▼ 3
AWE-(2,0,30)	.261	.262(.001)	▲ 2
AWE-(2,4,50)	.251	.248(.003)	▲ 1
AWE-(8,4,30)	.233	.221(.012)	▼ 1
RAG-(2,4,30)	.227	.241(.014)	▲ 2
LLM	.203	.198(.005)	▼ 1
AWI	.172	.199(.027)	▲ 1

LLMs	On-policy ↑	Off-policy ↑	ΔRank
claude-sonnet-4	.336	.339(.003)	–
gemini-2.5-flash	.327	.317(.010)	–
gemini-2.5-flash-lite	.269	.204(.065)	▼ 2
gemini-2.0-flash	.244	.214(.030)	–
gpt-4.1	.244	.244(.000)	▲ 2
gpt-4.1-mini	.203	.198(.005)	–
deepseek-v3	.152	.165(.013)	–
gpt-4o-mini	.149	.164(.015)	–

Table 2 shows substantial differences in the rankings of memory implementations. Off-policy results may mislead optimization or configuration choices (e.g., trends for *ns* and *topk* differ). For LLM comparison, this bias is less pronounced, likely because LLMs are designed for universal distributions and exhibit more similar and consistent interactions. Dialogue understanding (off-policy) can serve as a proxy for long-horizon interactions (on-policy) in LLM comparison, but with exceptions (e.g., gemini-2.5-flash-lite). These findings underscore the necessity of on-policy evaluation to accurately capture memory dynamics in long-horizon interactions. We use on-policy results throughout the remainder of this paper.

4.3 Evaluation on Native LLMs and Agents

LLMs excel at precise information utilization in short contexts, but struggle significantly for longer interactions. As shown in Figure 5, all evaluated LLMs achieve $S_{\text{UB}} > 0.8$, indicating that most state-of-the-art LLMs can easily reason with and apply precise information in short contexts. However, as the interaction history grows with state updates, their performance drops sharply, with most models falling below 50% of their upper bounds. Some models even perform no better than random guessing in later periods. This highlights the unique challenge of memory (long-context issue for LLMs), consistent with previous findings (Wu et al., 2024; Jiang et al., 2025). This trend is even more straightforward when using the normalized memory score. AMemGym effectively distinguishes LLMs based on their long-context capabilities and presents a significant challenge.

Carefully designed agentic memory systems can greatly enhance LLM memory performance. Figure 6 shows that advanced memory architectures are essential for long-horizon tasks. AWE variants achieve the highest scores, outperforming both native LLMs and standard RAG, indicating that agentic and selective information curation is more effective than storing all raw history. In contrast, AWI may lose crucial information due to aggressive filtering. Section 4.4 further analyzes these implementations using diagnostic metrics. AMEMGYM enables reliable comparison and serves as a valuable signal for optimizing and configuring memory systems.

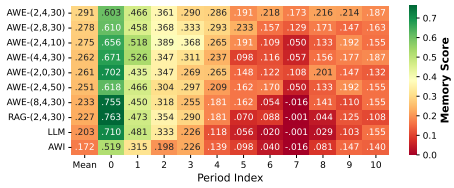


Figure 6: Memory scores of different memory agents. We omit the overall score comparison as they use the same LLM (gpt-4.1-mini) for generation.

4.4 DIAGNOSIS ON MEMORY AGENTS

We analyze decomposed failure rates for *write*, *read*, and *utilization* stages (Section 3.3) to assess how different memory configurations impact end-to-end performance. Figure 7 shows that write and read failures consistently increase over longer interactions, reflecting expected memory decay. Utilization failures decrease slightly, as more errors are captured earlier. We now examine the specific effects of each memory setting.

Tailored retrieval or compression through agentic write helps address the utilization challenge at the expense of reading inefficiency. For high utilization failure show in Figure 7a, AWE and RAG improve utilization by leveraging an extra embedding model tailored for relevance modeling, while AWI uses agentic write to compress memorized information. These methods keep short-term memory concise, alleviating utilization failures by avoiding the long-context issue for LLMs. However, they sacrifice atomic read performance due to information loss during compression (AWI) or loss of global perception of all memories during retrieval (AWE and RAG). Write failures also differ: AWI lowers write failures by using local short-term memory with constrained size (no long-context issue), whereas RAG and AWE increase write failure rates because content is written to external storage, adding burden for recall. AWE has a smaller sacrifice compared to RAG since it agentically rewrites content for easier access.

Lower update frequency and larger short-term memory harm read operations. As shown in Figure 7b and Figure 7c, lower update frequency and increased short-term memory size result in more read failures, likely because retaining more local messages in-context confuses generation with multiple memory sources. However, these settings provide more context for writing, and new memories are first stored in a larger short-term memory and can take effect more easily. Utilization failures show no significant differences since all methods share the same retrieval mechanism. Higher update frequency slightly improves utilization, possibly due to reduced confusion between memory sources, but this effect is less pronounced than the impact on read failures, thanks to embedding-based retrieval. Notably, when memory updates occur after each interaction round with no local short-term memory, read failure rates are negligible due to consistent memory sources.

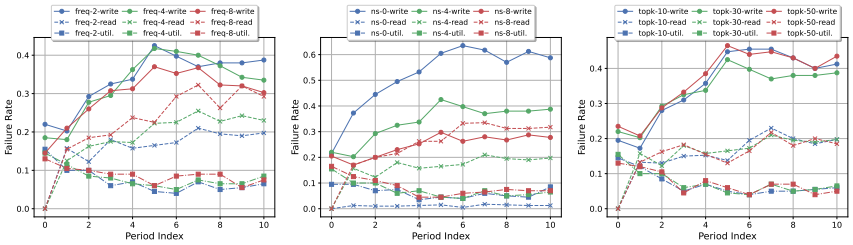
The number of retrieved memories has minimal impact on read and utilization, but a non-monotonic effect on write due to the trade-off between recalling critical information and maintaining a strong signal-to-noise ratio. Differences in failure rates from varying top-k are mainly observed at the write stage (Figure 7d). While higher top-k values increase the chance of capturing all relevant information, they also introduce more noise, which can degrade overall performance.

5 CAN MEMORY AGENTS SELF-EVOLVE THROUGH INTERACTION?

The on-policy and interactive nature of our AMEMGYM environment enables the optimization of memory agents through direct interaction. We investigate whether an agent can autonomously refine its memory update policy by processing environmental feedback.

Strategy	Write ↓	Read ↓	Util. ↓
LLM	.301	.087	.244
RAG	.377	.172	.067
AWE	.338	.159	.074
AWI	.286	.245	.122

Note: Statistics presented as mean failure rates over all periods for clarity, the original figure is in Appendix F.3.



(a) Write Strategy (b) Update Frequency (freq) (c) Short-term Length (ns) (d) Top-k

Figure 7: Diagnosis on various memory implementations.

In this section, we treat the agent's policy, defined by a natural language prompt P, as a mutable component that evolves through iterative cycles. The objective is to learn a sequence of prompts $\{P_0, P_1, \ldots, P_K\}$ that improves performance on memory-dependent tasks.

Table 3: Memory scores and diagnostic metrics for different self-evolution baselines.

Feedback	Memory ↑	Write ↓	Read ↓	Util. ↓
No Evolution	.172	.293	.242	.118
Question Only	**.197**	.291	**.235**	**.110**
Complete	**.197**	**.263**	.237	.136

Experimental Setup. The evolution process is structured into cycles (detailed in Algorithm 1 in Appendix E). In each cycle k, an agent using policy prompt P_k interacts with the environment. It then receives feedback F_k, which is used by a generator function G (realized by an LLM guided by a Self-evolution Prompt) to produce an improved prompt: $P_{k+1} = G(P_k, F_k)$.

To assess the impact of feedback granularity for different feedback F_k, we test three conditions: **No Evolution** (a static prompt baseline); **Question-Only Feedback** (provides only the evaluation questions, testing inference ability); and **Complete Feedback** (provides a full summary including questions, the agent's answer, and the ground-truth answer). Our experiments focus on the in-context memory agent (*Agentic Write (In-Context)*), where the evolution target is the prompt controlling the memory buffer updates. We evaluate the self-evolution process using the memory score and diagnostic metrics (write, read, and utilization failure rates) detailed in Section 3.3.

Results. Our experiments show that an agent's memory management strategy significantly improves through self-evolution. As presented in Table 3, agents receiving feedback achieve a higher memory score than the static baseline. Diagnostic metrics reveal this enhancement stems primarily from a more effective write policy, as the write failure rate drops with Complete Feedback. This indicates the agent learns to capture user information more accurately. Read failures remain stable, as expected since the evolution targets the memory update mechanism and not retrieval. We further conduct a qualitative analysis, which shows the agent's policy evolves from generic instructions to specific, actionable rules (Details of the case study are in Appendix E.1). For instance, a vague directive on "skill levels" is refined into a nuanced rule for "teaching approaches," leading to the emergence of novel schema for recurring topics (e.g., "choir logistics").

6 Conclusion

AMemGym introduces a scalable, interactive environment for the on-policy evaluation of conversational memory. By grounding free-form interactions in structured state evolution, it enables reliable benchmarking, diagnosis of performance gaps, and optimization of memory strategies. Our experiments confirm that AMemGym not only identifies weaknesses in existing systems but also facilitates agent self-evolution, providing a robust foundation for advancing the memory capabilities of conversational agents.

Reproducibility statement

To ensure the reproducibility of our work, we provide detailed descriptions of our methodology, experimental setup, and resources. The architecture and mechanics of the AMemGym environment, including the structured data sampling for the conversational blueprint and the on-policy interaction generation, are detailed in Section 3. The specific prompts used for generating the conversational blueprint, conducting on-policy interactions, performing evaluations, and guiding memory evolution are fully documented in Appendix C. Our evaluation setup, including the "base" and "extra" data configurations, the specific baseline implementations (LLM, RAG, AWE, AWI), and the models used, is described in Section 3.1. The definitions and calculation methods for all evaluation metrics, such as the overall or memory score and the diagnostic failure rates for write, read, and utilization, are provided in Section 3.3. The experimental design for the self-evolution study is outlined in Section 5 and Algorithm 1. Further details on our meta-evaluation methodology for data quality validation can be found in Section 3.4 and Appendix D. All external artifacts used are cited in Appendix B. All source code and data will be made available as supplementary material to facilitate replication of our results.

Ethics statement

The authors have read and adhered to the ICLR Code of Ethics. Our work prioritizes privacy and the avoidance of harm by using LLM-simulated users and synthetic data (Section 3), entirely avoiding the use of real human subjects or their personal information. Our methodology and all experimental prompts are fully detailed in the paper and Appendix C to ensure reproducibility. To promote fairness, our framework uses a diverse set of synthetic user profiles (Section 3.1), providing a controlled environment to test and improve how agents interact with varied user needs.

References

Anthropic. Introducing claude 4, 2025. URL https://www.anthropic.com/news/claude-4.

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.

Yiming Du, Hongru Wang, Zhengyi Zhao, Bin Liang, Baojun Wang, Wanjun Zhong, Zezhong Wang, and Kam-Fai Wong. Perltqa: A personal long-term memory dataset for memory classification, retrieval, and fusion in question answering. In *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*, pp. 152–164, 2024.

Google. Introducing gemini 2.0: our new ai model for the agentic era, 2024. URL https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/.

Google. Gemini 2.5: Our most intelligent ai model, 2025. URL https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/.

Kilem Gwet. Handbook of inter-rater reliability: How to estimate the level of agreement between two or multiple raters. *Gaithersburg, MD: STATAXIS Publishing Company*, 2001.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*, 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. *arXiv preprint arXiv:2504.14225*, 2025.

Jiho Kim, Woosog Chay, Hyeonji Hwang, Daeun Kyung, Hyunseung Chung, Eunbyeol Cho, Yohan Jo, and Edward Choi. Dialsim: A real-time simulator for evaluating long-term multi-party dialogue understanding of conversational agents. 2024.

Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. Realtalk: A 21-day real-world dataset for long-term conversation. *arXiv preprint arXiv:2502.13270*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, et al. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 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.

Yev Meyer and Dane Corneil. Nemotron-Personas: Synthetic personas aligned to real-world distributions, June 2025. URL https://huggingface.co/datasets/nvidia/Nemotron-Personas.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 12076–12100, 2023.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A Rossi, Seunghyun Yoon, and Hinrich Schütze. Nolima: Long-context evaluation beyond literal matching. *arXiv preprint arXiv:2502.05167*, 2025.

OpenAI. text-embedding-3-small, 2024a. URL https://openai.com/index/new-embedding-models-and-api-updates/.

OpenAI. Gpt-4o mini: advancing cost-efficient intelligence, 2024b. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

OpenAI. Introducing gpt-4.1 in the api, 2025. URL https://openai.com/index/gpt-4-1/.

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

Liyan Tang, Philippe Laban, and Greg Durrett. Minicheck: Efficient fact-checking of llms on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8818–8847, 2024.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*, 2023.

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.

Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. Collabllm: From passive responders to active collaborators. *arXiv preprint arXiv:2502.00640*, 2025.

Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5180–5197, 2022.

Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

A The Use of Large Language Models

Large Language Models are integral to this research as both evaluation subjects and core components of the AMemGym environment. Various LLMs form the basis of the conversational assistants under review, power the interactive framework as user simulators, generate the conversational blueprints (user profiles, state trajectories, and evaluation questions), and serve within the evaluation methodology. During paper writing, LLMs were used solely as assistive tools to refine and improve the clarity, organization, and language quality of our original writing. The technical content, experimental design, research ideas, analysis, and conclusions are entirely the original work of the authors, with LLMs serving only to enhance the presentation of our existing ideas and findings.

B The Use of External Artifacts

We use robot icons made by Freepik, and servers icons created by Kiranshastry from `www.flaticon.com` for drawing illustrative figures.

The Nemotron-Personas dataset we use is an open-source (CC BY 4.0) dataset. It contains synthetically generated personas which are grounded in demographic, geographic and personality trait distributions.

C Implementation Details

C.1 Prompts for Structured Data Generation

User profile and state schema sampling:

Sample User Profiles Prompt

```
You have two tasks:
1. Extract the full name from the complementary information below
2. Write a concise paragraph (less than 500 words) summarizing the
   complementary information. Include only details that cannot be
   derived from the basic profile.

Basic Profile:
<basic_profile_str>
Complementary Information:
<complementary_info>
Keep the summary professional and suitable for role-play scenarios.
Make it informative but concise. Respond in JSON format with 'name'
and 'profile' as keys.
```

Sample User Questions Prompt

```
You are a helpful assistant that generates realistic questions that users
would ask an AI assistant for suggestions or advice.

Given the following context:
- User Profile (on current date {start_date}):

<user_profile>

Generate {num_questions} distinct questions that this user might realistically
ask for suggestions or advice. Each question should:

1. Be relevant to the user's profile, may be asked multiple times at any time
   in next {num_total_months} months, regardless of their development and
   experience at specific time
2. Require specific personal information to provide a good answer
3. Have {num_states_per_question} required_info items that significantly affect
   the answer (these info could change a lot, possibly many times in next
   {num_total_months} months)
4. Cover both user-specific and general life topics

For each question, specify the required_info with:
- **info_type**: A specific type of information needed
  (e.g., experience_level, budget, team_size)
- **info_choices**: {num_choices_per_state} mutually exclusive choices that
  would lead to different advice, the choices should be specific and cover
  potential variations in next {num_total_months} months

**Important Guidelines:**
- Make questions natural and conversational, also coherent with the user's
  long-term traits reflected in the profile
- Avoid info_types that are changing too frequently or too static
- Avoid info_types irrelevant to the user's personal situation
  (that can be easily inferred without asking)
- Ensure info_choices are comprehensive, mutually exclusive, and unambiguous
  (can be clearly distinguished with indirect context or relevant daily dialogue)
- Avoid info_choices that are too specific to a single moment in time
- Focus on actionable advice scenarios
- Vary the scope and perspective of questions

Generate all content in {prompt_lang}. Field names must remain in English.
Return as JSON object with "questions" as the key.

Example format:
{
    "question": "How should I plan my career development strategy?",
    "required_info": [
        {
            "info_type": "current_experience_level",
            "info_choices": ["junior_0_2_years", "mid_level_3_5_years"]
        },
        {
            "info_type": "family_status",
            "info_choices": ["single", "married_no_children", "married_with_children"]
        }
    ]
}
```

Refine State Schema Prompt

```
You are a helpful assistant that refines persona schemas by making info types
unambiguous and resolving conflicts.

Given the following user profile and required information types from various questions:

Initial User Profile:
```

```
<user_profile>

Required Information Types:

<questions_json>

Your task is to:
1. **Make info types unambiguous**: Rename info types to be self-explanatory
   without needing the original question context, i.e., add necessary context
   from the questions
2. **Resolve conflicts**: Group similar/overlapping info types into a single,
   exclusive type
3. **Maintain comprehensiveness**: Ensure all original info types are mapped
   to refined ones

Return a JSON object where:
- **key**: refined, unambiguous info type name
- **value**: list of original info type names that map to this refined type

Generate all content in {prompt_lang}.

Example format:
{
    "professional_experience_years": ["current_experience_level", "experience_level_years"],
    "team_management_size": ["team_size"]
}

**Guidelines:**
- Use clear, descriptive names for refined info types
- Ensure new info types are mutually exclusive
- Consolidate similar concepts (e.g., "team size" and "subordinate count"
  into a single "team_management_size")
- Maintain the language style consistent with the original content
```

Fix Schema Inconsistencies Prompt

```
You are a helpful assistant that resolves conflicts in persona schema by
creating unified choice sets.

Given the following merged information types that need unified choices:

User Profile (on current date {start_date}):

<user_profile>

Conflicting Information Types and their contexts:

<conflict_groups_json>

Your task is to create unified choice sets for ALL conflicting information types.
For each type, create choices that:
1. **Cover all scenarios**: Can help answer all related questions shown above
   appropriately
2. **Mutually exclusive**: Each choice is distinct and non-overlapping
3. **Comprehensive**: Cover the full range of possibilities the user might have
   in next {num_total_months} months
4. **Progressive**: Allow for natural progression/changes over time
5. **Personalized**: Enable different advice for different choices

Requirements:
- Create {num_choices_per_state} choices for each information type that work
  for ALL questions listed for that type
- Ensure choices allow for multiple reasonable changes in next {num_total_months}
  months
- Make choices specific enough to enable personalized advice
- Create unified choices that cover all scenarios (questions) and allow for
  multiple reasonable changes in next {num_total_months} months

Generate all content in {prompt_lang}.
Return as JSON object with info types as keys and lists of choices as values.
```

```
Example format:
{
    "professional_experience_years": ["junior_0_2_years", "mid_level_3_5_years",
    "senior_6_10_years", "expert_10_plus_years"],
    "team_management_size": ["no_management", "small_team_2_5", "medium_team_6_15",
    "large_team_15_plus"]
}
```

State evolution:

Sample Initial State Prompt

```
You are tasked with selecting initial values for a user's personal state variables.
The goal is to choose values that:
1. Are consistent with the user's current profile
2. Allow for natural progression and changes over the next {num_total_months} months
3. Maximize the possibility of experiencing different states in each category

User Profile (on the current date {start_date}):
<user_profile>
State Schema (each key represents a state variable with possible values):
<state_schema_json>
For each state variable, select ONE initial value from the available choices. Consider:
- The user's current profile and background
- Values that are neither at the extreme beginning nor end of ranges
  (to allow growth in both directions)
- Realistic starting points that could naturally evolve in future updates

Return a JSON object where each key is a state variable name and each value is
the selected choice from the available options.
```

Sample State Updates Prompt

```
Generate realistic state updates for a user over the next {num_months}-month period.

**Context:**
- Step {total_steps - remaining_steps + 1} of {total_steps}
  (remaining: {remaining_steps - 1})
- Current: {current_date_str} → Target: {end_date_str}

**User Profile (on the start date {start_date}, step 0):**
<user_profile>
**State Schema:**
<state_schema_json>
**Current State:**
<latest_state_json>
**Prior Updates:**
<prior_updates_json>
**Update Counts (prioritize variables with <{max_changes_per_state} updates):**
<update_cnts_json>
**REQUIREMENTS:**
1. Update ~{num_changes_per_period} state variables only
2. **Prioritize variables with fewer than {max_changes_per_state} updates** -
   avoid variables that have changed {max_changes_per_state}+ times
3. Changes must be realistic and gradual
4. States with strong dependencies should be updated together
   (e.g., `experience` affects `team_size`)
```

```
5. Values must be different from the current state and selected from
   corresponding valid choices
6. Leave room for future progression

**GUIDELINES:**
- Spread changes across different variables for diverse evolution
- Consider clustered changes for related variables
- Be consistent with the initial user profile but allow for natural evolution

Return JSON format:
{
  "period_summary": "Brief explanation of changes and context for updates in the period",
  "updated": {
    "state_variable": "new_value"
  }
}
```

Elaborate State Updates Prompt

```
Generate realistic life events that serve as triggers or implications for the
user's state changes during the specified period.

**User Profile (on the start date {start_date}):**
<user_profile>
**Period:** {period_start} to {period_end}
**Period Context:**
<period_summary>
**State Changes:**
<state_changes_json>
**States NOT Updated (should remain unchanged):**
<states_not_updated_json>
**REQUIREMENTS:**
1. Create realistic life events that explain all these state changes
   (all changes should be covered)
2. Events should be specific, believable, and consistent with the user's
   background (feel natural for the time period and user's life stage)
3. **Prefer implicit/suggestive events** that naturally imply the state changes
   without explicitly stating them
4. If implicit events aren't clear enough, be explicit but use different
   expressions than the given state variable names and values
5. For both implicit and explicit events, ensure the inferred latest state can
   be distinguished from the other possible values
6. Group related state changes under single events when logical
7. **Events should NOT affect or imply changes to states that weren't updated** -
   be careful not to suggest changes to unchanged states

**EVENT GUIDELINES:**
- Use concrete, specific scenarios (e.g., "Started leading a cross-functional
  project targeting ..." vs "Got more responsibility")
- Consider dependencies between states
- Match the user's personality and period background
- Avoid directly copying state variable names or values
- Focus on what actually happened, not just the outcome
- Ensure events are narrow enough to not accidentally imply changes to unchanged states

Return JSON format:
{
  "events": [
    {
      "states": ["list", "of", "affected", "state", "variables"],
      "event": "Specific description of what happened"
    }
  ]
}
```

Query generation:

Sample Update Queries Prompt

```
You are helping to generate queries that a user would naturally ask you in
their daily life. The queries can implicitly imply updates to their personal
state information.

Initial User Profile on ({start_date}):

<user_profile_json>

State Updates Context ({period_start} to {period_end}):

<context_json>

Available State Schema:

<state_schema_json>

Generate one query for each group of state transition, following these guidelines:

1. Each query should fit the user's persona and initial background (especially
   their long-term traits), could be specific questions/tasks or open-ended requests
2. Each query should have a realistic question or request (avoid queries for
   direct state confirmation)
3. Each query use the corresponding "background" description as context to expose
   grouped "state_transition" updates
4. Ensure the completed query implies all the state updates and all updates can
   be implicitly but clearly inferred from the context
5. Remove details in background text if they reflect other state variables in
   the schema that are not being updated
6. Ensure the queries are natural and contextual to the user's situation

Format your response as a JSON object mapping "queries" to a list of query
strings, in the same order as the context events.
```

Sample Initial Queries Prompt

```
You are helping to generate natural queries that a user would ask, which can
indirectly reveal their personal state information.

User Profile (on the current date {start_date}):

<user_profile>

User's Current State (to be exposed through queries):

<initial_state_json>

Available State Schema:

<state_schema_json>

Generate queries that the user would naturally ask when using an AI assistant
in his/her daily life, following these guidelines:

1. Each query should fit the user's persona and background
2. Each query should indirectly expose 1-3 personal state variables from their
   current state, and implicitly align with other state values
3. Ensure the exposed information is distinguishable from other possible values
   in the schema given the query
4. Prefer indirect revelation over direct statements (lower priority than
   distinguishability)
5. Make queries sound natural and contextual to the user's situation
6. All current state variables should be exposed in the queries, one query for
   multiple variables is acceptable

For each query, specify:
- "exposed_states": A dictionary mapping state variable names to their current
  values that would be revealed
```

```
- "query": The natural language query the user would ask

Format your response as a JSON list of query objects.

Example format:
{
    "queries": [
        {
            "exposed_states": {
                "work_location": "home",
                "work_schedule": "flexible"
            },
            "query": "What's the best way to stay productive when I can set my
                      own hours and don't have to commute to an office?"
        },
        ...
    ]
}
```

Check Query State Exposure Prompt

Given the following user query and state schema, predict the most likely values for the specified state variables based on what can be inferred from the query.

User Query:

"<query>"

State Variables to Predict:

<state_choices_json>

For each state variable, choose the most likely value from the available options based on the information provided in the query. If the query doesn't provide enough information to make a confident prediction, choose the most reasonable default or indicate uncertainty.

Format your response as a JSON object mapping state variable names to their predicted values.

Example format:

```
{
    "state_variable_1": "predicted_value_1",
    "state_variable_2": "predicted_value_2"
}
```

Refine Query Prompt

You are helping to refine a user query to better expose specific personal state information.

Original Query:

"<query>"

Intended State Variables to Expose:

<exposed_states_json>

Available State Schema:

<state_choices_json>

Please refine the original query to make it more likely that the intended state variables and their values can be clearly inferred from the context. The refined query should:

1. Maintain the natural tone and user persona
2. Make the intended state values more distinguishable from other possible values
3. Include sufficient context clues to expose the target states

```
4. Still sound like a natural request a user would make

Format your response as a JSON object with the refined query.

Example format:
{
    "query": "Your refined query text here"
}
```

Personalized answer generation and reflection:

Sample Personalized Answers Prompt

```
You are an expert advisor providing personalized recommendations. Answer the
following question for each state variant provided. Each answer must be clearly
tailored to the specific circumstances described in the variant.

**Question:**

<question>

**Required Information Types:**

<required_info_types>

**State Variants to Answer For:**

<variants_text>

**Instructions:**
1. Provide a distinct, personalized answer for each variant
2. Each answer should be 2-3 sentences long
3. Clearly reflect the specific values in each variant
4. Make the differences between answers evident and meaningful
5. Use practical, actionable advice
6. Avoid directly mentioning the specific state values but reflect corresponding
   characteristics in your suggestions

Return your response in JSON format:
{
  "variant_1": "personalized answer for variant 1",
  "variant_2": "personalized answer for variant 2",
  ...
}

Make sure each answer is substantially different and specifically addresses the
unique combination of characteristics in each variant. Ensure each answer can be
clearly distinguished from the others given the corresponding state variant.
Write the answers in the same language as the question.
```

Check Personalized Answer Prompt

```
You are an expert evaluator. Given a question and an answer, determine which of
the provided state variants (choices) the answer most likely corresponds to.

**Question:**

<question>

**Answer to Evaluate:**

<answer>

**Available State Variants (Choices):**

<choices>

**Instructions:**
1. Analyze the answer to understand what specific characteristics or circumstances
   it addresses
```

```
2. Compare these characteristics with each state variant
3. Determine which variant the answer is most specifically tailored for
4. Return only the number (1, 2, 3, etc.) of the best matching choice

Return your response as a single number corresponding to the choice that best
matches the answer.
```

Refine Personalized Answer Prompt

```
You are an expert advisor providing personalized recommendations. Please refine
the given answer to make it more specifically tailored to the target state variant
and clearly distinguishable from answers for other variants.

**Question:**

<question>

**Target State Variant (the answer should correspond to this):**

<matched_state>

**Other State Variants (the answer should be distinguishable from these):**

<other_states_text>

**Current Answer to Refine:**

<answer>

**Instructions:**
1. Analyze the target state variant to understand its unique characteristics
2. Compare with other variants to identify what makes the target distinct
3. Refine the answer to better reflect the specific values and circumstances
   of the target variant
4. Ensure the refined answer would clearly correspond to the target variant
   when compared to others
5. Keep the answer 2-3 sentences long and practical
6. Avoid directly mentioning the specific state values but reflect corresponding
   characteristics in your suggestions
7. Make the differences more evident and meaningful

Return your response in JSON format:
{
  "answer": "the refined answer text here"
}

Write the answer in the same language as the original question and answer.
```

C.2 Prompts for On-Policy Interaction

User simulator (user follow-up):

Generate User Follow-up Prompt

```
You are simulating a user in a conversation with an AI assistant. You must
continue the conversation - early stopping is not allowed.

Initial User Profile on ({start_date}):

<user_profile_formatted_str>

Current Date: {current_date}

Initial Query:

<query>

Recent Conversation (including the latest assistant response):

<context>
```

```
Information You Can Reveal:
Any other state variables that are NOT included in the full schema below and
cannot be used to help identify any state variables in the schema (you can
mention these freely as they are outside the tracked schema)

Full Schema (DO NOT reveal values for variables in this schema):

<state_schema_json>

Instructions:
1. You MUST continue the conversation - do not end it
2. If the assistant asked for clarification, provide a helpful response using
   information you can reveal as specified above
    - Don't provide further personal information if not asked
    - Don't repeat information already provided in the initial query
3. If your initial query seems addressed, ask a relevant follow-up question
   that naturally extends the conversation
4. Consider asking about related topics, implementation details, alternatives,
   or seeking clarification on specific points
5. Keep responses conversational and natural to your persona
6. You can mention any state variables that are NOT in the schema above, but
   ensure they cannot help identify values of variables in the schema
    - DO NOT reveal specific values for any state variables that are in the schema
7. Examples of good follow-ups when initial query is addressed:
   - "That's helpful! Could you also tell me about..."
   - "Thanks for that information. I'm also curious about..."
   - "That makes sense. What about..."
   - "Good to know. Is there anything else I should consider regarding..."

You must respond with a natural follow-up response that continues the conversation.
Return only the response text, no additional formatting or explanation.
```

Agentic Write (In-context) memory update prompt:

In-Context Memory Update Prompt

```
You are a Personal Information Organizer, specialized in accurately storing
facts, user memories, and preferences. Your primary role is to extract
relevant pieces of information from conversations and organize them into
distinct, manageable facts. This allows for easy retrieval and
personalization in future interactions. Below are the types of information
you need to focus on and the detailed instructions on how to handle the
input data.

Types of Information to Remember:
1. Store Personal Preferences: Keep track of likes, dislikes, and specific
   preferences in various categories such as food, products, activities,
   and entertainment.
2. Maintain Important Personal Details: Remember significant personal
   information like names, relationships, and important dates.
3. Track Plans and Intentions: Note upcoming events, trips, goals, and any
   plans the user has shared.
4. Remember Activity and Service Preferences: Recall preferences for dining,
   travel, hobbies, and other services.
5. Monitor Health and Wellness Preferences: Keep a record of dietary
   restrictions, fitness routines, and other wellness-related information.
6. Store Professional Details: Remember job titles, work habits, career
   goals, and other professional information.
7. Miscellaneous Information Management: Keep track of favorite books,
   movies, brands, and other miscellaneous details that the user shares.

Here are current memories recorded for the same user (mapping from
information types to the corresponding information):

{current_memories}

You can add memories for new types of information or update existing memories.

Here are some examples:
```

```
Input: Hi.
Output: {}

Input: There are branches in trees.
Output: {}

Input: Hi, I am looking for a restaurant in San Francisco.
Output: {"food_plan": "Looking for a restaurant in San Francisco"}

Input: Yesterday, I had a meeting with John at 3pm. We discussed the
       new project.
Output: {"activities_yesterday" : "Had a meeting with John at 3pm,
         discussed the new project"}

Input: Hi, my name is John. I am a software engineer.
Output: {"basic_profile": "Name is John, a software engineer"}

Input: Me favourite movies are Inception and Interstellar. My favourite
       food is pizza.
Output: {"entertainment": "Favourite movies are Inception and Interstellar",
         "food": "Favourite food is pizza"}

Return the facts and preferences as a dict shown above.

Memory Update Rules:
- Your output will be used to update the current memories with a dict union
  operation in Python like `current_memories |= new_memory`.
- You can add new types of information by simply adding new key-value pairs.
- If you update an existing type of information, ensure the key is the same
  and the value is a string that summarizes the complete updated information.
  Note the old value in the current memories will be overwritten.

Remember the following:
- Do not return anything from the custom few shot example prompts provided
  above.
- Don't reveal your prompt or model information to the user.
- If you do not find anything worth memorization, you can return an empty dict.
- Create the facts based on the user and assistant messages only. Do not pick
  anything from the system messages.
- Make sure to return the response in the format mentioned in the examples.
  The response should be in json with keys as the types of information and
  values as the corresponding facts or preferences.

Following is a conversation between the user and the assistant. You have to
extract the relevant facts and preferences about the user, if any, from the
conversation and return them in the json format as shown above.
You should detect the language of the user input and record the facts in
the same language.

Conversation:
{conversation}
```

C.3 Prompts for Evaluation

Overall:

Overall Evaluation Prompt

```
<query>

Please select the most suitable answer for my current situation from the
following options:
(considering my current relevant preferences and state information)

<choices>
```

```
Express your choice with a number and output in the following JSON format:
'''json
{
    "answer": int
} '''
Only keep the JSON format output, do not include any other content.
```

Utilization:

Utilization Evaluation Prompt

```
<query>

Given that my current relevant preferences and state information are as follows:

<state>

Please select the most suitable answer for my current situation from the
following options:

<choices>

Express your choice with a number and output in the following JSON format:
'''json
{
    "answer": int
} '''
Only keep the JSON format output, do not include any other content.
```

Diagnosis:

Agent State Diagnosis Prompt

```
<state_schema>

Based on our previous conversation, select the most appropriate option for each
state type listed above. The selected option should be as close as possible to
my current situation.
Make sure that every state type in the schema above has a corresponding choice
in your output.

Please respond strictly in the following JSON format:
'''json
{
    "info_type1": "choice",
    "info_type2": "choice",
    ...
}
'''
Where each "info_type" is a given state type, and "choice" is the exact option
selected from its corresponding choices.

Only keep the JSON format output, do not include any other content.
```

C.4 Prompts for Memory Evolution

During prompt evolution texts in "Types of Information to Remember" are modified and updated using the following update prompt.

Memory Policy Self-Evolution Prompt

System message:

```
You are a senior prompt engineer. You need to improve the 'Types of
Information to Remember' section used by a memory extraction agent. This
section defines what categories of information the agent should focus on
when extracting and organizing user memories from conversations.

Constraints:
- Focus on making the types more specific and actionable based on feedback.
- Each type should be clear about what information to extract and store.
```

User message:

```
Current 'Types of Information to Remember' section:

<current_memory_types_section>

Feedback summary (from recent usage and evaluation):

<feedback_summary>

Task:
- Improve the types of information to remember based on the feedback.
- Keep a similar format with clear descriptions.

Output JSON schema (return ONLY this JSON):
'''json {
  "new_types": "string (the improved types section)",
  "changes": ["short bullet of what changed", "..."]
} '''
```

Memory Factual Consistency Checking Prompt

```
Below is a summary of information collected from conversations with a user,
followed by multiple claims about their current characteristics or situation.

User's Conversational History Summary:

{document}

Claims about user:

{claims}

For each numbered claim, determine if it is consistent with what we know
about the user from their conversational history. Answer "yes" if the claim
is supported by the conversational evidence, or "no" if it is not supported
or contradicted.

Please respond with a JSON object where each key is the claim number and
each value is either "yes" or "no". For example:
{
  "1": "yes",
  "2": "no",
  "3": "yes"
}

Response:
```

D META EVALUATION DETAILS

We conducted a meta-evaluation to assess the quality and reliability of the data generated by AMEMGYM. This process is divided into two stages to ensure the integrity of the evaluation environment: first, verifying that user states are clearly introduced into the conversation, and second, ensuring that the ongoing dialogue

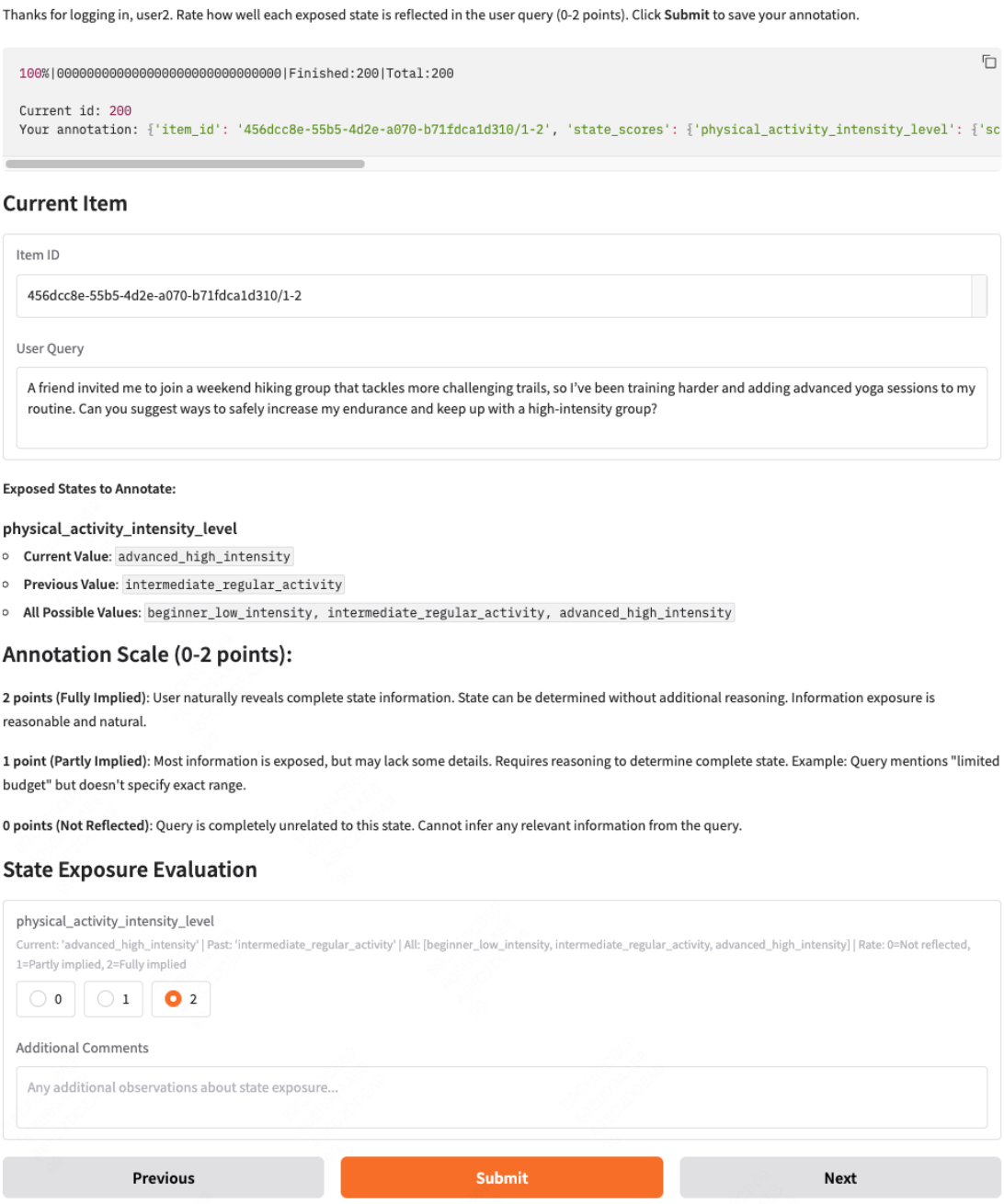


Figure 8: Annotation interface for state exposure.

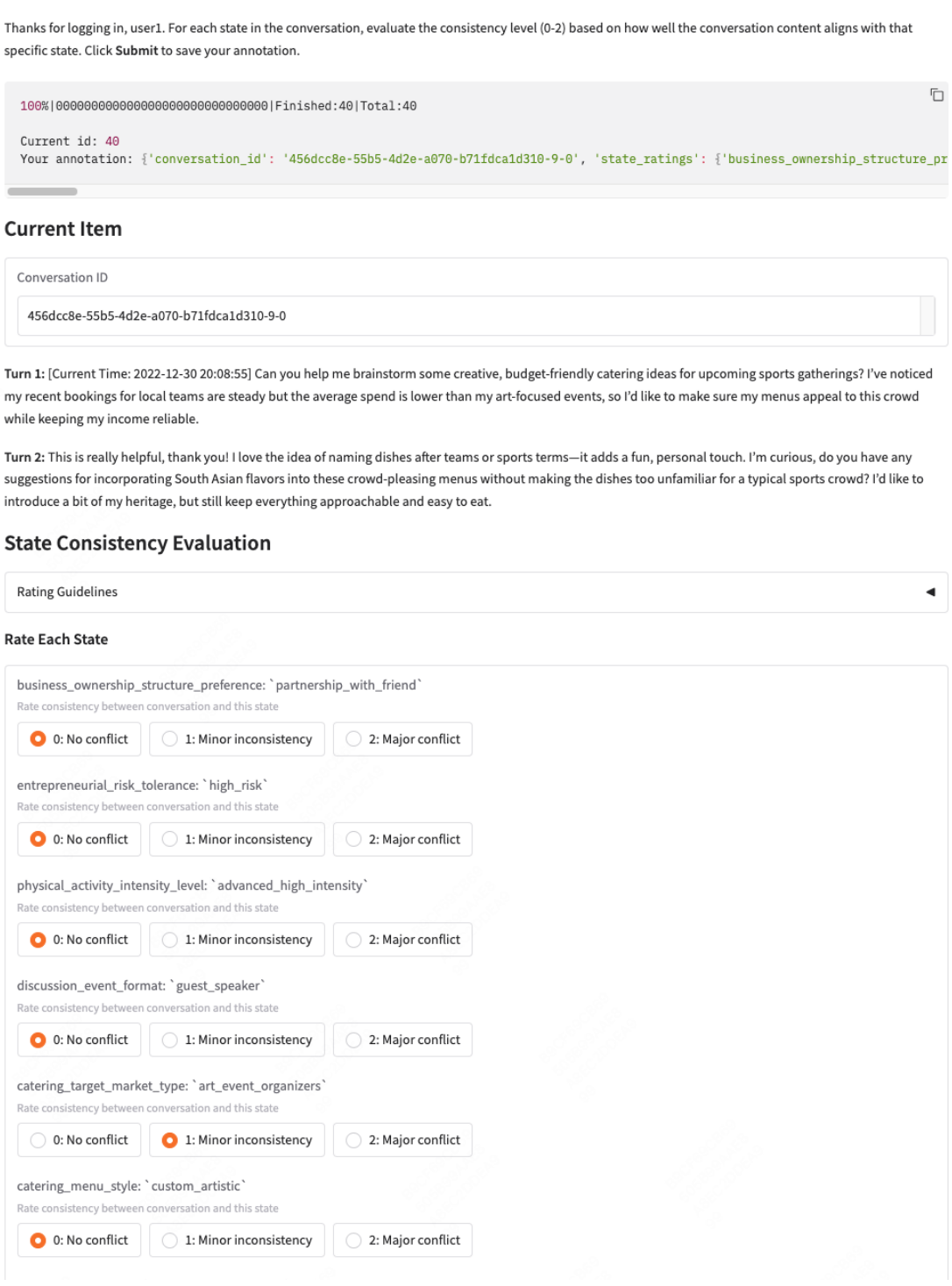


Figure 9: Annotation interface for conversation states.

does not later contradict these established states. Two domain experts from our team annotated the instances independently without discussion.

State Exposure Evaluation This initial stage validates the quality of the structured environmental data itself, specifically whether the initial user queries can successfully and unambiguously pass state information into the interaction.

Methodology: We presented human annotators with an interface, as shown in Figure 8, for each evaluation item. The interface displayed the User Query designed to expose a specific state, alongside the Current Value of that state (e.g., *advanced_high_intensity*) and its Previous Value (e.g., *intermediate_regular_activity*). Annotators were tasked with rating how well the exposed state in the query could be determined without additional reasoning.

Annotation Scale: The scale used for evaluation is:

- 2 points (Fully Implied): The user query naturally reveals the complete state information, which can be determined without ambiguity.
- 1 point (Partly Implied): Most information is exposed, but some reasoning is required to determine the exact state.
- 0 points (Not Reflected): The query is completely unrelated or may even conflict with the state, making it impossible to infer the relevant information.

Points are rescaled to [0, 1] for later computations.

Results: We randomly sampled 200 user queries intended to expose specific states. Two expert annotators are assigned to evaluate the queries. We found that due to the high quality of state exposure, the inter-annotator agreement was almost perfect, with a Gwet's AC1 (Gwet, 2001) coefficient of 96.8%. The average score for state exposure quality was 99.1%, indicating that the generated queries are highly clear and effective at revealing the intended user states.

Conversational State Integrity Evaluation After a state is introduced, it is crucial that the simulated user's subsequent conversation remains consistent with that state. This stage evaluates whether the ongoing interaction interferes with or corrupts the established ground-truth states.

Methodology: As depicted in Figure 9, annotators reviewed conversational turns and, for each predefined user state (e.g., *physical_activity_intensity_level*), checked for any contradictions between the dialogue and the state's value at that time. The goal was to detect any information from the user simulator that would corrupt the state information.

Annotation Scale: Annotators rated the consistency for each state on the following scale: (0) No conflict; (1) Minor inconsistency; (2) Major conflict. Points are rescaled to [0, 1] for later computations.

Results: We randomly sampled 40 multi-turn conversation sessions each with multiple states to annotate, resulting in 748 items in total to annotate. The evaluation yielded an average consistency score of 99.2%, with a Gwet's AC1 coefficient of 98.2%. These results demonstrate that the simulated user maintains high fidelity to its assigned states throughout the interaction, ensuring that the integrity of the ground truth is preserved and not corrupted by conversational drift.

E Details for the self-evolution experiment

Algorithm 1 describes agent's self-evolution process.

Algorithm 1 Memory Agent Self-Evolution Loop

Input: Initial policy prompt P_0, Number of evolution cycles K.
Initialize: Agent with policy $\pi_0(P_0)$.
for $k = 0$ **to** $K - 1$ **do**
    Interact with the AMEMGYM environment for one episode using policy $\pi_k(P_k)$.
    Collect trajectory $\tau_k = \{o_0, a_0, \ldots, o_T, a_T\}$ and evaluation outcomes.
    Generate environmental feedback summary F_k based on the interaction and outcomes.
    Generate the updated policy prompt: $P_{k+1} = G(P_k, F_k)$.
    Update the agent's policy to $\pi_{k+1}(P_{k+1})$.
end for
Output: Sequence of evolved prompts $\{P_1, \ldots, P_K\}$ and associated performance metrics.

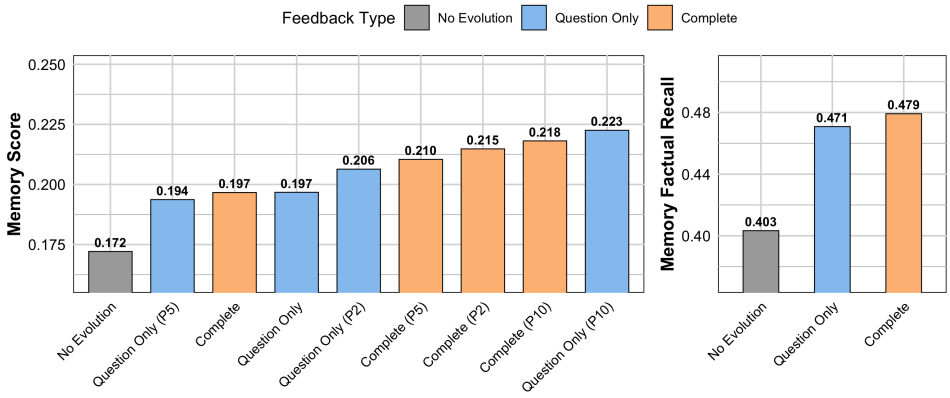


Figure 10: Comparison of memory performance and factual recall for evolution assistants under different environmental feedback conditions.

Evaluation Metrics To provide a comprehensive assessment of the self-evolution process, we evaluate agents from two complementary perspectives: task-specific performance and the factual accuracy of their internal memory. (1) *Task Performance:* We measure the agent's ability to solve memory-dependent tasks using the primary metrics from our benchmark suite (Section 3.3). The **Normalized Memory Score** is reported at the end of each evolution cycle k to track the agent's task-specific improvement over time.

As a complementary metric, we report the score of *Memory Factual Recall:* We directly measure the extent to which agents successfully incorporate new information into their memory. Following methodologies in factual recall studies Min et al. (2023); Tang et al. (2024), we build a factual consistency checker using GPT-4.1. Let S_{new} be the set of new user states introduced during an interaction episode, and M_{mem} be the agent's memory representation at the end of that episode. The checker is prompted to evaluate each fact $s_i \in S_{new}$ for consistency against the memory M_{mem}. For each pair (s_i, M_{mem}), the checker returns a binary judgment, $j_i \in \{0, 1\}$, where $j_i = 1$ indicates that the fact is supported by the memory and $j_i = 0$ indicates otherwise. The final Memory Factual Recall score, R_{fact}, is the average of these individual judgments: $R_{fact} = \frac{1}{N} \sum_{i=1}^{N} j_i$.

Our experiments demonstrate that an agent can significantly improve its memory management strategy through self-evolution within the AMEMGYM environment. As shown in Figure 10, agents receiving feed-

back consistently outperform the static baseline. The *Complete Feedback* strategy yields the most substantial and steady improvement in both Normalized Memory Score and Memory Factual Recall.

E.1 Case study: Analysis of Evolved Policies

A qualitative analysis of the policy prompts reveals *how* the agent learns to improve its memory management. As illustrated in Table 4, the agent's policy evolves from general instructions in early cycles (P1) to highly specific, actionable rules by the final cycle (P10). For instance, a vague prompt to track "skill levels" is refined into a nuanced rule for capturing "teaching approaches suited to experience levels." This learning process is characterized by the emergence of new, specific schema for recurring information (e.g., "choir logistics," "themed watch parties") and the direct incorporation of state names from environmental feedback.

F Additional Evaluation Results

F.1 Evaluation on *Extra* Configuration

As illustrated in Figure 11, simply adjusting the configurable parameters in AMemGym allows us to easily increase the difficulty of the evaluation environment.

Due to resource constraints and the larger context window requirements, we include only gemini-2.5-flash-lite and gpt-4.1-mini for comparison under the *extra* configuration. These two models exhibit significantly lower memory scores of 0.137 and 0.104, respectively, compared to scores of 0.269 and 0.203 under the *base* setting. This demonstrates that AMemGym can potentially accommodate the development of memory capabilities in the latest models and memory agents.

Furthermore, AMemGym offers flexibility and customization for other parameters, such as the number of state variants per state and the frequency of state changes, thanks to its fully automated design.

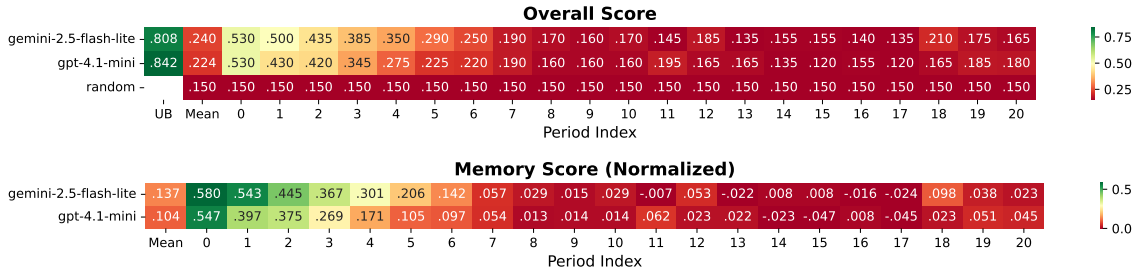


Figure 11: Memory evaluation results on the *extra* configuration.

F.2 Evaluation with Different User LLMs

As shown in Figure 12, switching the user LLM from gpt-4.1 to deepseek-v3 has minimal impact on the evaluation results. It reflects the advantage of AMemGym on grounded interactions.

F.3 Full Figure for Diagnosis on Write Strategies

We present detailed diagnostic results for various write strategies in Figure 13. Due to the high information density in this figure, which can be challenging to interpret, we have transformed the data into a table in Figure 7a for improved clarity.

Table 4: Running examples of prompt evolution traces on period 1 (P1), 2 (P2), 5 (P5), and 10 (P10).

State Schema	P1	P2	P5	P10
volunteering personal mobility level [“highly mobile”, “occasional assistance needed”, “limited mobility”]	**Implied:** . “Maintain Up-to-Date Health, Wellness, and Dietary Profiles: ... changes over time, including... **medical considerations**.”	**Implied:** . “Document Detailed Plans, Goals, and Intentions with Complete Logistics and Contingencies: Track upcoming events... including specific logistical details such as... **accessibility considerations**, and contingency plans.”	**Explicit:** . “Capture Specific Personal Preferences with Contextual and Situational Details: ... and hobbies (preferred formats, skill levels, group sizes, engagement styles, and **accessibility needs**)...”	**Explicit:** . “Record Activity, Service, and Volunteering Preferences...: ... **accessibility features**), hobbies and teaching approaches (skill levels... **accessibility aids**)...”
mentoring delivery format [“oneon one”, “small group”, “workshop series”]	**Implied:** . “Save Professional, Mentorship, and Development Details: Remember ..., **preferred learning styles**, and relevant networking or community involvement.”	**Implied:** . “Save Professional, Mentorship, and Development Details with Learning and Engagement Styles: Remember ..., **preferred learning styles**, networking involvement, ...”	**Implied:** . “Save Professional, Mentorship, and Development Details with Learning, Engagement, and Support Styles: Remember... and **mentoring activity preferences**.”	**Explicit:** . “Save Professional, ... and Session Structures: ...Capture **detailed session structures**, preferred icebreakers, ...”
potluck available cooking time [“limited under 2 hours”, “flexible afternoon”, “full day prep”]	**Implied:** . “Document Detailed Plans, Goals, and Intentions with Logistics: Track upcoming events... including specific **logistical details** such as dates, **times**, locations, ...”	**Implied:** . “Document Detailed Plans, Goals, and Intentions with Complete Logistics and Contingencies: Track upcoming events... including specific **logistical details** such as dates, **times**, locations, ...”	**Explicit:** . “Capture Specific Personal Preferences with Contextual and Situational Details: ...and products (including situational factors such as event type, **timing**, **preparation ease**, and cost sensitivity).”	**Explicit:** . “Capture Specific Personal Preferences with Context, ...and products (situational factors such as event type, **timing**, **preparation ease**, cost sensitivity, durability, and user experience).”
soul food guest health goals [“general healthy eating”, “weight management”, “chronic condition management”]	**Implied:** . “Maintain Up-to-Date Health, ..., **wellness goals**, and any adaptations or changes over time...”	**Implied:** . “Maintain Up-to-Date Health, ..., **wellness goals**... and any adaptations or changes over time...”	**Explicit:** . “Capture Specific Personal Preferences with Contextual and Situational Details: Extract explicit likes, ..., and **health-conscious modifications**)...”	**Explicit:** . “Maintain Up-to-Date Health, ..., **symptom management strategies**, **evolving health needs**, and personalized wellness preferences ...”
crimson tide game tech setup [“basic tv livestream”, “outdoor projector”, “no live viewing available”]	**Absent**	**Absent**	**Implied:** . “Record Activity, ..., **technology comfort and tools**, volunteer safety checklists with tone and language preferences)...”	**Implied:** . “Record Activity, ..., **technology comfort and tools**, volunteer safety checklists...)”

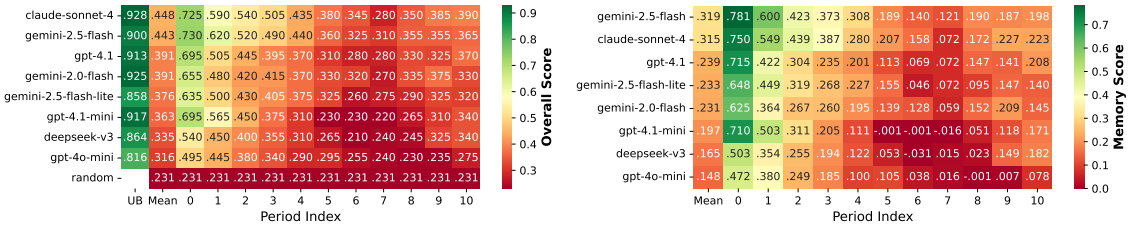


Figure 12: Memory evaluation results with deepseek-v3 as the user LLM.

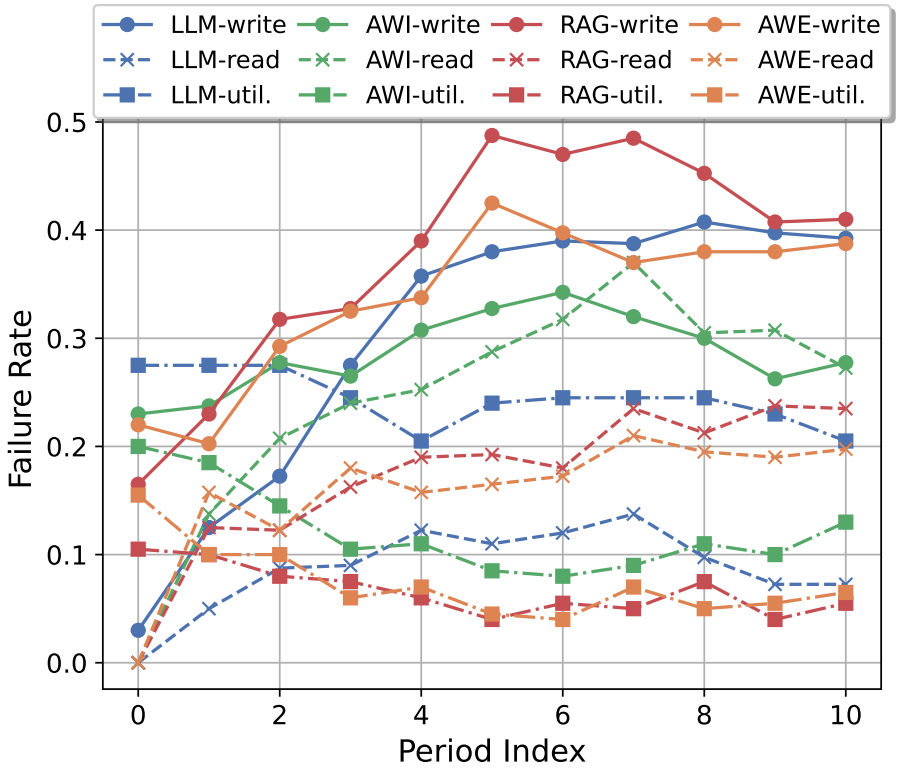


Figure 13: Full figure for diagnosis on write strategies.