EvoMemEval: Evaluating Large Language Models at Evaluating Memory Utility in Self-Evolving Agents

Anonymous ACL submission

Abstract

Self-evolving memory enables LLM agents to accumulate experience from past interactions and reuse it for future task solving. As memory is updated online, agents need to assess which memories should be reused, revised, or discarded. Such utility is ultimately grounded in execution. This creates a practical need to predict memory utility before execution. While recent LLM-as-a-judge studies suggest that LLMs can serve as automatic evaluators, it remains unclear whether they can reliably judge the execution utility of self-evolved memories. To address this gap, we introduce **EvoMemEval**, a benchmark for testing large language models on memory utility evaluation in self-evolving agents. Our benchmark preserves the sequential nature of memory construction: each memory snapshot is first built through online evolution within a task stream, and memory items are then evaluated on held-out future tasks by comparing executions with and without the memory. We formulate both point-wise utility prediction and pair-wise memory comparison. Experiments on four agentic benchmarks with frontier LLMs show that current evaluators remain weak: point-wise macro-F1 is close to the random baseline, and pair-wise comparison is only modestly better. Further analyses show that retrieval similarity is not a reliable proxy for utility, memory form strongly affects judging difficulty, and planning before judging can improve performance. These results establish EvoMemEval as a concrete testbed for studying memory utility evaluation in self-evolving agents.

1 Introduction

LLM-based agents increasingly operate in sequential task environments, where they must interact with external environments, learn from trial-and-error, and reuse prior experience in future decisions (Shinn et al., 2023; Wang et al., 2023, 2024; Suzgun et al., 2025; Wei et al., 2025; Xu et al., 2025). A central mechanism for this capability is self-evolving memory: the agent records trajectories from environment interaction, distills them into reusable experience, retrieves relevant memories for later tasks, and updates its memory state online. In this sense, memory serves as the bridge between past trajectories and future agent behavior, enabling agents to improve at test time without updating their model parameters (Shinn et al., 2023; Wang et al., 2023; Suzgun et al., 2025).

Recent self-evolving memory systems instantiate this bridge in diverse forms, including raw trajectories, few-shot examples, reflective lessons, procedural tips, adaptive cheatsheets, workflows, knowledge bases, skills, and reusable tools or APIs (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023; Packer et al., 2023; Wang et al., 2024; Suzgun et al., 2025; Tang et al., 2025; Zhang et al., 2025). These memories differ substantially in abstraction level, structure, granularity, and update mechanism. Across this diversity, the central criterion remains the same: a memory is valuable only insofar as it improves future task execution. Yet this value is not captured by surface relevance. A memory may closely match the task description while lacking actionable information for the agent, or even steering the agent toward an ineffective strategy (Xiong et al., 2025; Zhang et al., 2026). Conversely, a concise memory with limited semantic overlap may encode a transferable procedure that improves execution.

Despite the importance of memory utility, current evaluation mainly focuses on agent-level outcomes. Existing benchmarks and systems typically measure end-to-end task success, memory ability, or cumulative performance (Wu et al., 2024; Wei et al., 2025; Hu et al., 2026). Such evaluations are necessary for comparing complete systems, but they provide only coarse feedback: they reveal whether a memory framework works on average,

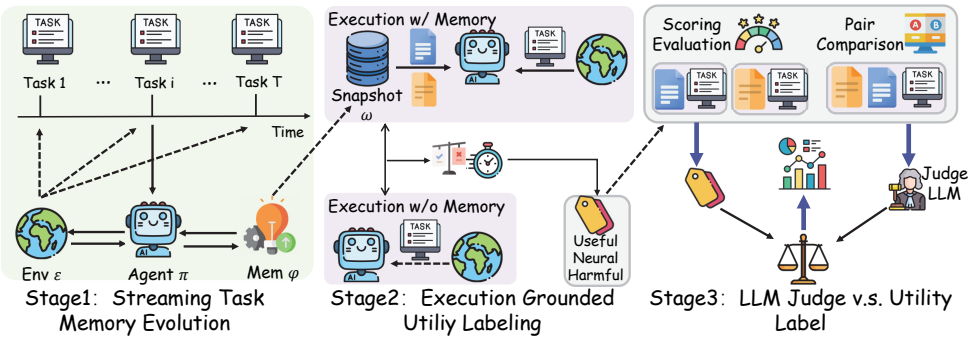

Figure 1: Overview of EvoMemEval. Stage 1 streams source tasks through an agent and a self-evolving memory system, producing memory snapshots from online interaction. Stage 2 freezes a snapshot and derives execution-grounded utility labels by comparing the same evaluation task with and without a candidate memory. Stage 3 evaluates LLM judges in point-wise scoring and pair-wise comparison settings against the retained utility labels.

not whether a particular memory is actually worth using for the current task. As memory systems become more dynamic and self-evolving, this missing memory-level supervision becomes a practical bottleneck.

This naturally creates a practical need to predict memory utility before execution. Recent LLM-as-a-judge studies suggest that LLMs can serve as automatic evaluators for complex outputs (Zheng et al., 2023; Zeng et al., 2024; Chen et al., 2024), but it remains unclear whether they can reliably judge the execution utility of self-evolved memories before the agent acts. In this work, we introduce **EvoMemEval**, a benchmark for testing LLMs on memory utility evaluation in self-evolving agents. Our benchmark follows the lifecycle of practical self-evolving memory systems. For each construction setting, we instantiate a memory framework with a memory base model and pair it with a task-solving agent backed by a task-solving base model. The agent then processes a sequential source task stream, allowing the memory state to evolve through real online task execution. The resulting memory state is then frozen, and candidate memories are retrieved for held-out future tasks, following the standard use of retrieval to select non-parametric context for language models and agents (Karpukhin et al., 2020; Lewis et al., 2020; Packer et al., 2023). For each task–memory pair, we run the same agent on the same task with and without the memory, and label the memory by the resulting change in task success and cost.

EvoMemEval is designed around three principles. First, it is execution-grounded: memories are produced by realistic self-evolving memory frameworks during online task solving with real agentic environments, and labels are derived from controlled executions rather than manual preference annotation. Second, it is diverse: benchmark instances cover multiple agentic environments, memory frameworks, and memory base models. Third, it emphasizes label stability: utility labels are retained only when controlled executions produce consistent labels across repeated trials and evaluation base models. These choices allow EvoMemEval to study memory-level utility evaluation while preserving the realistic origin of self-evolving memories.

We formulate two evaluation settings. In the point-wise setting, the evaluator receives a task and one memory, and predicts its execution utility. In the pair-wise setting, the evaluator receives a task and two candidate memories, and selects the memory with higher expected utility. Both settings test whether current LLMs can provide fine-grained supervision for memory use before the agent executes the task.

Experiments on four agentic benchmarks show that memory utility prediction remains challenging for current frontier LLMs. Even strong models struggle to distinguish memories that improve execution from those that only appear relevant, suggesting that memory utility evaluation requires more than semantic relevance estimation or generic response-quality judgment. These results identify memory utility assessment as a missing capability for reliable self-evolving agents and establish EvoMemEval as a concrete testbed for studying it.

Our contributions are summarized as follows:

- **Benchmark.** We introduce EvoMemEval, a benchmark for evaluating whether LLMs can assess the execution utility of individual memories produced by self-evolving agents. Its construc-

tion preserves online memory evolution and derives stable memory-level labels from controlled executions.

- **Evaluation.** We evaluate strong LLMs across diverse agentic environments, memory frameworks, and memory forms under both point-wise utility prediction and pair-wise memory comparison settings.
- **Findings.** (1) We find that current LLM evaluators remain weak: point-wise macro-F1 is close to the random baseline, pair-wise comparison is only modestly better, (2) retrieval similarity is not a reliable proxy for utility, (3) memory form strongly affects judging difficulty, (4) and planning before judging can improve performance.

2 Related Work

Self-evolving memory for LLM agents. External memory has become a common mechanism for enabling agents to accumulate information and reuse prior experience over time. Early agent systems used memory-like mechanisms such as reflective feedback, virtual context management, and skill libraries to support learning without weight updates (Shinn et al., 2023; Packer et al., 2023; Wang et al., 2023). Recent systems differ in what they store and how they use it, including workflow memories (Wang et al., 2024), distilled tips and adaptive cheatsheets (Suzgun et al., 2025), reflective lessons (Zhao et al., 2024), knowledge-base style memories (Tang et al., 2025), mobile-agent memories (Wang et al., 2025), and broader self-evolving frameworks such as Evo-Memory, MemEvolve, and MemRL (Wei et al., 2025; Zhang et al., 2025, 2026). Recent analysis further shows that memory management can induce experience-following behavior, where agents over-rely on retrieved experiences even when they are not useful for the current task (Xiong et al., 2025). These methods study how memory should be written, structured, selected, and maintained, while our goal is to evaluate whether individual memories actually help future task solving.

LLM-as-a-judge. Another relevant line of work asks whether large models can serve as reliable judges for quality assessment, pairwise preference comparison, and response evaluation (Zheng et al., 2023; Zeng et al., 2024; Chen et al., 2024). Reward modeling is also central to preference-based learning and RLHF, where models learn from pairwise human judgments or ranked outputs (Christiano et al., 2017; Ouyang et al., 2022). Closely related reward-model benchmarks evaluate whether RMs can capture human preferences, subtle response-quality differences, or agent trajectory quality (Lambert et al., 2024; Liu et al., 2024; Lù et al., 2025; Men et al., 2025). These benchmarks have shown that strong LLMs can sometimes approximate human judgments, but also suffer from substantial weaknesses such as position bias, verbosity bias, and inconsistency. Our setting differs from standard LLM-as-a-judge and RM evaluation in two ways. First, the object being evaluated is not a response, answer, or full trajectory, but an individual memory whose value is determined by later task-solving outcomes. Second, our labels are grounded in downstream execution outcomes rather than human preference alone. This makes our benchmark closer to decision-time memory utility modeling than to conventional output evaluation.

Memory Evaluation. Recent benchmarks have also begun probing long-term memory and memory supervision more directly. StreamBench, Evo-Memory, EvoMemBench, and MemoryAgentBench evaluate continual improvement, self-evolving memory, and incremental multi-turn memory in sequential agent settings (Wu et al., 2024; Wei et al., 2025; Hu et al., 2026), while MemoryRewardBench studies whether reward models can evaluate long-term memory management in LLM systems (Tang et al., 2026). These efforts are complementary to ours. StreamBench, Evo-Memory, EvoMemBench, and MemoryAgentBench mainly assess full agent adaptation or memory ability over time, while MemoryRewardBench focuses on reward-model assessment for memory-management trajectories. Our benchmark instead targets a more localized but practically critical question: given a task and an individual memory item, can an evaluator infer the memory's utility before execution? This memory-level utility perspective is precisely the gap that EvoMemEval is designed to fill.

3 EvoMemEval

3.1 Benchmark Overview

EvoMemEval evaluates whether LLM evaluators can assess the utility of individual memories produced by self-evolving agents. Each instance pairs a future task with a memory and labels the

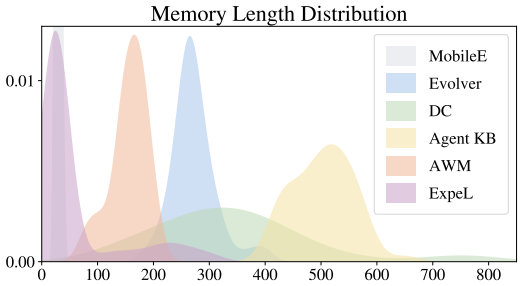

Figure 2: Length distribution of memories across different frameworks. Vertical axis: density. Horizontal axis: length;

memory by its observed effect on task execution. EvoMemEval is built around three principles:

- **Execution-grounded and dynamic.** Memories are produced by real self-evolving memory frameworks during online task execution, rather than manually written or offline synthesized. Labels are also grounded in actual executions, obtained by comparing the same task with and without the memory.
- **Diverse agent-memory settings.** EvoMemEval covers multiple agentic environments, memory frameworks, memory representations, memory base models, and task-solving base models. This diversity reflects the heterogeneous forms in which self-evolving memory appears in practical agent systems.
- **Stable utility labels.** We retain a memory label only when controlled executions induce the same utility label across repeated trials and different execution base models.

3.2 Problem Setup

Following prior work on self-evolving memory (Wei et al., 2025; Zhang et al., 2025), we consider a sequential source task stream $\mathcal{T}_{\text{src}} = \{t_k\}_{k=1}^{K}$. Each memory construction setting is denoted by

$$z = (\Psi, \Omega_\phi, \pi_\rho) \in \mathcal{Z}, \tag{1}$$

where Ψ is the task environment, Ω_ϕ is a memory framework Ω instantiated with memory base model ϕ, and π_ρ is the task-solving agent backed by base model ρ.

To make the memory operations explicit, we denote the instantiated memory system as

$$\Omega_\phi = (\mathcal{E}_\phi, \mathcal{U}_\phi, \mathcal{R}, \mathcal{G}_\phi), \tag{2}$$

where $\mathcal{E}_\phi$, $\mathcal{U}_\phi$, $\mathcal{R}$, and $\mathcal{G}_\phi$ denote experience extraction, memory update, retrieval, and memory management, respectively. Concrete memory frameworks may implement only a subset of these operations, but they share the same high-level lifecycle of retrieving prior memory, solving the current task, extracting new experience, and updating the memory state (Wang et al., 2024; Suzgun et al., 2025; Zhao et al., 2024; Wu et al., 2025).

Under setting z, the memory state evolves from $M_0^z = \varnothing$. Before task t_k, the memory system provides the agent with memory context

$$c_k^z = \mathcal{R}(M_{k-1}^z, t_k). \tag{3}$$

The agent then interacts with the environment and produces a trajectory conditioned on retrieved memory

$$\tau_k^z \sim \pi_\rho(\cdot \mid t_k, c_k^z; \Psi). \tag{4}$$

The memory system distills this trajectory by extracting an experience unit and updating the memory state:

$$\epsilon_k^z \in \mathcal{E}_\phi(\tau_k^z), \quad M_k^z = \mathcal{U}_\phi(M_{k-1}^z, \epsilon_k^z). \tag{5}$$

After all K source tasks are processed, we freeze M_K^z for utility annotation on held-out tasks.

3.3 Benchmark Construction

We build EvoMemEval in two stages. The first stage accumulates memory through a source task stream. The second stage freezes the resulting memory state and annotates candidate memories on held-out future tasks.

Stage I: Streaming Task Memory Evolution. For each construction setting $z \in \mathcal{Z}$, the agent processes $\mathcal{T}_{\text{src}}$ sequentially under memory system Ω_ϕ. The memory state is updated online according to the evolution process described above, following prior memory-agent systems that accumulate reflections, workflows, skills, or experience summaries from interaction traces (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2023, 2024). This stage preserves the realistic origin of self-evolving memory: memories are extracted from task execution trajectories and accumulated by the real memory framework.

Stage II: Execution Grounded Utility Labeling. We then construct benchmark instances from held-out tasks $\mathcal{T}_{\text{eval}} = \{q_i\}_{i=1}^{M}$. For each task $q \in \mathcal{T}_{\text{eval}}$, the memory system retrieves a candidate memory set from snapshot M_K^z:

$$\begin{aligned} \mathcal{C}_q^z &= \mathcal{R}(M_K^z, q) \\ &= \{m_{q,1}^z, m_{q,2}^z, \ldots, m_{q,r}^z\}. \end{aligned} \tag{6}$$

For each memory item $m \in \mathcal{C}_q^z$, we run the same agent on the same task with and without m, and derive a utility label from the resulting difference in task success and cost. This follows the preference-learning view that utility should be grounded in comparative outcomes, while adapting it from human preference over trajectories to observed memory effects in agent execution (Christiano et al., 2017; Ouyang et al., 2022; Lù et al., 2025).

More formally, let $o^+(q, m; \pi_{\rho'})$ and $o^-(q; \pi_{\rho'})$ denote the outcomes of the two executions with the same evaluation agent $\pi_{\rho'}$, where $+$ indicates the run using memory m and $-$ indicates the run without it. A deterministic comparator g maps the outcome pair to a utility label:

$$y_{\rho'}(q, m) = g\big(o^+(q, m; \pi_{\rho'}), o^-(q; \pi_{\rho'})\big). \quad (7)$$

Here $y_{\rho'}(q, m)$ is the utility label induced by evaluation agent $\pi_{\rho'}$. The comparator first compares task success, and uses token cost as an efficiency signal only when both runs solve the task reliably. We retain a benchmark instance only when repeated runs and different evaluation agents induce the same label; the retained point-wise label is denoted as $y_s(q, m)$. This produces stable labels in $\{\texttt{useful}, \texttt{neutral}, \texttt{harmful}\}$ while keeping each memory grounded in a realistic self-evolving process.

Constructed Data Instances. The construction produces two forms of evaluator instances: We denote the point-wise label space as $\mathcal{Y}_\text{s} = \{\texttt{useful}, \texttt{neutral}, \texttt{harmful}\}$ and the pair-wise label space as $\mathcal{Y}_\text{p} = \{\texttt{A}, \texttt{B}, \texttt{tie}\}$. For pair-wise comparison, we form memory pairs only within the same candidate set. We define $\mathcal{P}_q^z = \{(m_a, m_b) : m_a, m_b \in \mathcal{C}_q^z,\ m_a \neq m_b\}$. Thus, both memories in a pair are selected for the same task under the same construction setting $z = (\Psi, \Omega_\phi, \pi_\rho)$, matching the practical setting where a deployed memory system ranks candidates returned by its own selection procedure (Lewis et al., 2020; Karpukhin et al., 2020).

- **Point-wise evaluation:** triples (q, m, y_s), where $m \in \mathcal{C}_q^z$ and $y_s \in \mathcal{Y}_\text{s}$. The evaluator e_θ receives (q, m) and predicts y_s.
- **Pair-wise comparison:** tuples (q, m_a, m_b, y_p), where $(m_a, m_b) \in \mathcal{P}_q^z$ and $y_p \in \mathcal{Y}_\text{p}$. The evaluator e_θ receives (q, m_a, m_b) and predicts y_p. Pair labels are derived from point-wise labels under the order $\texttt{useful} \succ \texttt{neutral} \succ \texttt{harmful}$; pairs with the same point-wise label are assigned $\texttt{tie}$.

Benchmark Statistics. Starting from candidate memories produced by realistic memory systems, repeated-run and cross-agent consistency filtering retains 769 out of 2199 candidate instances. The stable pool remains neutral-heavy, with `useful`, `neutral`, and `harmful` accounting for 13%, 68%, and 19% of retained instances, respectively. We therefore construct evaluation splits balanced by label, resulting in 300 point-wise instances and 282 pair-wise instances. Each subset has balanced label categories. Detailed statistics are provided in Appendix A.1.

4 Experiments

In this section, we evaluate leading LLMs as memory utility evaluators on EvoMemEval under our unified evaluation pipeline, focusing on four key research questions (RQs):

- **RQ1:** How reliably do current LLMs evaluate memory utility?
- **RQ2:** Can retrieval similarity explain memory utility?
- **RQ3:** What label preferences do current evaluators exhibit?
- **RQ4:** How does memory form affect judging difficulty?

We further include a small prompt analysis and a case study to examine why static memory utility judgment remains difficult.

4.1 Experimental Setup

Benchmarks. We evaluate EvoMemEval on four representative agentic benchmarks, including **ALFWorld** (Shridhar et al., 2021), **BabyAI** (Chevalier-Boisvert et al., 2019), **ScienceWorld** (Wang et al., 2022), and **WebShop** (Yao et al., 2022). These environments cover household task completion, language-conditioned navigation, scientific interaction and reasoning, and web-based shopping decisions, respectively.

Evaluators. Because no model is specifically trained for memory utility evaluation in self-evolving agents, we evaluate a set of frontier LLMs as general-purpose evaluators, following recent LLM-as-a-judge and reward-model studies (Zheng et al., 2023; Lambert et al., 2024; Liu et al., 2024). Our current evaluation spans the Claude, Kimi (Team et al., 2025b), GLM (Team et al., 2025a), DeepSeek (DeepSeek-AI et al., 2025), MiniMax, Qwen (Yang et al.,

Models	*ALFWorld*		*BabyAI*		*ScienceWorld*		*WebShop*		*Avg.*	
	Score	**Pair**	**Score**	**Pair**	**Score**	**Pair**	**Score**	**Pair**	**Score**	**Pair**
Proprietary Models										
Claude-Sonnet-4.6	42.01	49.00	27.50	**43.80**	**45.56**	38.30	28.60	38.10	**35.92**	**42.30**
Qwen3-Max	40.50	40.00	28.40	33.50	35.50	37.90	29.80	35.10	33.50	36.60
Open-source Models										
Kimi-K2.5	39.10	48.30	36.00	35.20	36.90	36.00	27.00	39.00	34.80	39.60
GLM-5	41.10	48.90	29.80	35.40	36.20	**46.30**	28.20	32.90	33.80	40.90
DeepSeek-V4-Flash	25.10	44.70	32.40	34.10	36.20	41.10	**31.90**	**39.20**	31.10	39.80
DeepSeek-V4-Pro	37.30	**50.60**	34.40	35.20	26.60	37.70	27.00	33.90	31.30	39.40
MiniMax-M2.7	33.60	47.50	31.10	35.10	33.20	41.40	26.90	37.40	31.20	40.30
MiMo-V2.5-Pro	38.90	45.40	37.30	30.10	32.00	33.80	16.90	33.50	30.80	35.70
Qwen3.5-Plus	**42.70**	41.50	28.30	36.30	25.40	38.00	30.00	30.20	31.60	36.50
Qwen3-235B-A22B-Instruct	33.10	42.50	31.20	35.30	37.40	33.90	28.90	36.10	32.70	36.90
Qwen3-30B-A3B-Instruct	27.67	45.31	**39.26**	31.92	35.20	36.09	27.24	37.80	32.34	37.78
Qwen3-14B	33.61	46.41	29.56	33.10	34.80	30.86	26.15	30.60	31.03	35.24
Qwen3-4B	30.00	40.30	30.50	29.70	22.80	30.80	22.40	35.00	26.40	34.00

Table 1: Main results on EvoMemEval using macro-F1. Each subset is evaluated under either the score setting or the pair comparison setting with ties. Averages are reported separately for point-wise score prediction and pair-wise comparison.

Model	ALFWorld	BabyAI	Avg.
Claude-Sonnet-4.6	42.01	27.50	34.76
w/ Few-shot	43.26	26.80	35.03
w/ Plan	**45.62**	**27.81**	**36.72**
DeepSeek-V4-Flash	25.10	32.40	28.75
w/ Few-shot	37.20	28.90	33.05
w/ Plan	**37.37**	**37.56**	**37.47**

Table 2: Prompt analysis on point-wise memory utility prediction using macro-F1. The base row uses the direct prompt; w/ Plan refers to PLAN THEN JUDGE.

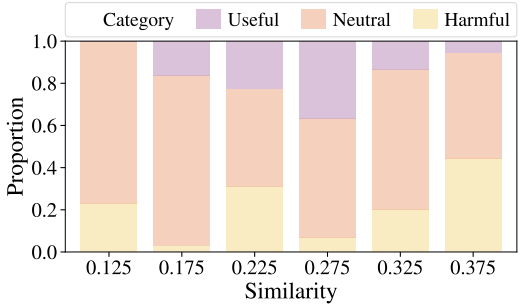

Figure 3: Utility label distribution across retrieval similarity buckets.

2025), and MiMo families. We report results for Claude-Sonnet-4.6, Qwen3-Max, Kimi-K2.5, GLM-5, DeepSeek-V4-Flash, DeepSeek-V4-Pro, MiniMax-M2.7, Qwen3.5-Plus, Qwen3-235B-A22B-Instruct, Qwen3-30B-A3B-Instruct, MiMo-V2.5-Pro, Qwen3-14B, and Qwen3-4B. We report macro-F1 for both point-wise and pair-wise settings. Since each setting has three balanced labels, uniform random prediction gives an expected macro-F1 of approximately 33.33%.

Memory Frameworks. We construct benchmark memories using representative self-evolving memory frameworks, including AGENT WORKFLOW MEMORY (AWM) (Wang et al., 2024), AGENT-KB (Tang et al., 2025), EXPEL (Zhao et al., 2024), DYNAMIC CHEATSHEET (Suzgun et al., 2025), EVOLVER (Wu et al., 2025), MOBILE-E (Wang et al., 2025). These frameworks span several styles of memory abstraction, including workflow memories, distilled tips and shortcuts, reflective experience summaries, and structured knowledge bases.

Benchmark Instantiation. We use 15 source tasks to build each memory snapshot with Gemini-3.1-Flash-Lite and Claude-Haiku-4.5 as memory base models ϕ and 15 disjoint held-out tasks for evaluation. We use the same two models as task-solving base models ρ for π_ρ. During utility annotation, for every evaluation task, we retrieve the top-3 candidate memories, and each paired execution with and without memory is repeated three times.

4.2 Analysis of Results (RQ1)

Current evaluators remain weak under direct memory utility prediction. Table 1 reports macro-F1 on the score and pair settings. Since each setting has three balanced labels, uniform random predic-

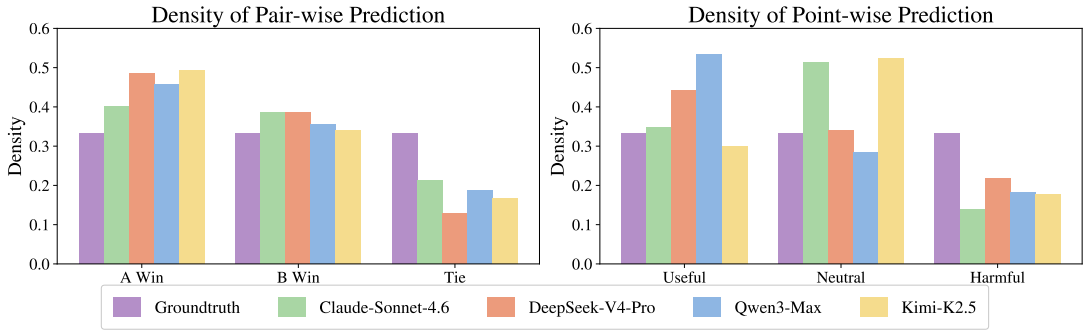

Figure 4: Gold and predicted label distributions in the score and pair settings.

tion gives an expected macro-F1 of approximately 33.33%. In the score setting, most models are close to this baseline, and nine out of thirteen models fall below it. Claude-Sonnet-4.6 obtains the best average score macro-F1 at 35.92, followed by Kimi-K2.5 at 34.80. This shows that judging whether a single memory is `useful`, `neutral`, or `harmful` remains difficult for current LLM evaluators.

Pair comparison is easier, but still far from reliable. Pair-wise evaluation gives higher macro-F1 than score prediction for most models. Claude-Sonnet-4.6 obtains the best pair average at 42.30, followed by GLM-5 at 40.90 and MiniMax-M2.7 at 40.30. However, even the best pair result is less than 10 points above the random baseline. Thus, comparing two memories is easier than assigning an absolute utility label, but current evaluators still do not provide reliable memory-level preference signals.

Environment structure changes the judging difficulty. The point-wise setting is especially difficult on WebShop, where every model is below the random baseline and the best result is 31.90. ALFWorld is more favorable for point-wise evaluation, with Claude-Sonnet-4.6, GLM-5, and Qwen3.5-Plus all above 41.0. ScienceWorld shows a different pattern: Claude-Sonnet-4.6 reaches 45.56, but several other models fall below 30. These differences suggest that memory utility judgment depends on the structure of the underlying agent environment, not only on the evaluator family.

4.3 Planning Before Judging

The main evaluation gives each evaluator only the task and a candidate memory, so the model must infer the future execution process from static text. As a diagnostic, we first add the trajectory from a run without memory to the judge input. On ALFWorld with Claude-Sonnet-4.6, this raises macro-F1 from 42.0 to 51.1. This result suggests that execution evidence can help the evaluator identify where a memory might change the outcome. Although this setting is not practical for memory utility evaluation, since it requires running the task before judging the memory, it provides a useful insight: memory utility judgment should be grounded in an anticipated execution process. Motivated by this, we propose **PLAN THEN JUDGE**, a prompt variant that asks the evaluator to first sketch the likely task solving process and then assign the memory utility label.

Table 2 shows that planning before judging improves over the direct prompt in all four tested settings. For Claude-Sonnet-4.6, PLAN THEN JUDGE improves ALFWorld by 3.61 points and gives a smaller gain on BabyAI, raising the average from 34.76 to 36.72. For DeepSeek-V4-Flash, the gains are larger: 12.27 points on ALFWorld, 5.16 points on BabyAI, and 8.72 points on average. Few-shot prompting helps ALFWorld but hurts BabyAI for both models, while PLAN THEN JUDGE shows the best average performance in both models. These results support the same conclusion as the trajectory diagnostic: **memory utility judgment benefits from reasoning about the execution process, but a generated plan is still an imperfect substitute for observing an actual rollout**.

5 Analysis

5.1 Analysis of Relevance *vs.* Utility (RQ2)

More similar memories are not necessarily more useful. Figure 3 groups memories by retrieval similarity and shows the retained utility labels in each group. Useful memories are most concentrated in the middle similarity range, while the most similar group contains many harmful memories. In our current statistics, the [0.25, 0.30) bucket contains the largest useful proportion, whereas the [0.35, 0.40] bucket has visibly fewer useful memories

Memory Framework	Memory Form	Claude-Sonnet-4.6	Kimi-K2.5	DeepSeek-V4-Flash	Avg.
ExpeL	Insight / case summary	35.77	32.11	32.18	33.35
Mobile-E	Tip	**46.80**	**50.30**	**42.65**	**46.58**
AWM	Workflow summary	28.66	22.37	23.15	24.73
Dynamic Cheatsheet	Procedural summary	22.23	25.85	22.60	23.56
Agent-KB	Structured knowledge	35.23	33.86	25.81	31.63
EvolveR	Guiding principle	33.16	28.65	35.72	32.51

Table 3: Point-wise macro-F1 by memory form.

and many more harmful ones. This pattern explains why retrieval score alone is a weak filter for memory use. Similarity can recover memories that mention related objects, goals, or workflows, as in standard retrieval-augmented systems (Karpukhin et al., 2020; Lewis et al., 2020), but it does not distinguish a transferable strategy from a misleading prior run. The difficult cases are not unrelated memories; they are memories that look relevant but encode the wrong constraint, wrong subgoal, or wrong search behavior.

5.2 Prediction Bias (RQ3)

Models make different label choices even on the same evaluation split. Figure 4 compares the gold label distribution with model predictions. Claude-Sonnet-4.6 and Kimi-K2.5 place many point examples into `neutral`, while Qwen3-Max favors `useful`. Harmful memories are less consistently detected across models. In the pair setting, the displayed models predict fewer `tie` labels than the gold distribution, especially DeepSeek-V4-Pro and Kimi-K2.5, which more often force a winner between two memories.

5.3 Difficulty by Memory Form (RQ4)

Different memory frameworks expose different forms of memories to the evaluator. Table 3 reports average point-wise macro-F1 for representative evaluators under each form. **Concise tips are easier to judge than long workflow summaries.** Mobile-E tips obtain the highest average macro-F1, 46.58, and are the easiest form for all three evaluators. In contrast, AWM workflow summaries and Dynamic Cheatsheet procedural summaries are the hardest, with averages of 24.73 and 23.56. This gap suggests that more detail does not necessarily make memory utility easier to evaluate. Longer summaries can expose useful execution evidence, but they also carry source-task assumptions, intermediate actions, and constraints that may look transferable while changing the target task in the wrong way. Table 4 gives a concrete example of this difficulty. The evaluator recognizes a plausible WebShop search heuristic, but misses that relaxing search constraints directly conflicts with the target task's hard dietary requirements.

Field	Content
Task	Find a non-alcoholic and zero sugar mocktail under $50.00 in WebShop.
Memory	A self-evolved summary from a Dell desktop search recommends starting broad and then removing one attribute at a time when results are sparse.
Prediction	`neutral`. The evaluator treats the memory as generic search advice, not task-specific guidance.
Label	`harmful`.
Error	The memory is unsafe because non-alcoholic and zero sugar are hard requirements, not optional attributes. Following it may relax exactly the constraints that determine task success.

Table 4: A representative point-wise error. The table shows a condensed judge input and an excerpt of the self-evolved memory summary. Blue spans mark hard constraints in the target task, while red spans mark the memory heuristic that conflicts with them. The example illustrates a common evaluator error: judging a memory by whether it sounds like reasonable task advice, rather than whether it improves the concrete execution.

6 Conclusion

We introduced EvoMemEval, a benchmark for testing whether LLMs can evaluate the utility of memories produced by self-evolving agents. EvoMemEval preserves the memory lifecycle by building memory snapshots through online task execution and assigning labels through controlled runs with and without memory. Experiments show that current LLM evaluators remain unreliable: point-wise prediction is close to random, pair-wise comparison is only modestly stronger, and judging difficulty depends on both environment structure and memory form. These results highlight memory utility evaluation as a concrete bottleneck for reliable self-evolving agents.

7 Limitations

EvoMemEval currently covers four representative agentic environments and several self-evolving memory frameworks. While this provides diverse sources of tasks, memory forms, and online memory updates, future work can further scale the benchmark to more environments, longer task streams, and broader memory-system designs, especially in real-world deployed agents. In addition, this work focuses on benchmark construction and evaluation rather than training a specialized memory utility evaluator. A natural next step is to use the collected utility signals to train models for memory selection, filtering, revision, and online self-evolution.

References

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yinuo Liu, Yaochen Wang, Huichi Zhou, Qihui Zhang, Yao Wan, Pan Zhou, and Lichao Sun. 2024. Mllm-as-a-judge: Assessing multimodal llm-as-a-judge with vision-language benchmark. *Preprint*, arXiv:2402.04788.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. Babyai: A platform to study the sample efficiency of grounded language learning. *Preprint*, arXiv:1810.08272.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2026. Evaluating memory in llm agents via incremental multi-turn interactions. *Preprint*, arXiv:2507.05257.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2024. Rewardbench: Evaluating reward models for language modeling. *Preprint*, arXiv:2403.13787.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. 2024. Rm-bench: Benchmarking reward models of language models with subtlety and style. *Preprint*, arXiv:2410.16184.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. 2025. Agentrewardbench: Evaluating automatic evaluations of web agent trajectories. *Preprint*, arXiv:2504.08942.

Tianyi Men, Zhuoran Jin, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2025. Agent-rewardbench: Towards a unified benchmark for reward modeling across perception, planning, and safety in real-world multimodal agents. *Preprint*, arXiv:2506.21252.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. *Preprint*, arXiv:2010.03768.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2025. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, Ge Zhang, Jiaheng Liu, Xingyao Wang, Sirui Hong, Chenglin Wu, Hao Cheng, Chi

Wang, and Wangchunshu Zhou. 2025. Agent kb: Leveraging cross-domain experience for agentic problem solving. *Preprint*, arXiv:2507.06229.

Zecheng Tang, Baibei Ji, Ruoxi Sun, Haitian Wang, Wangjie You, Yijun Zhang, Wenpeng Zhu, Ji Qi, Juntao Li, and Min Zhang. 2026. Memoryrewardbench: Benchmarking reward models for long-term memory management in large language models. *arXiv preprint arXiv:2601.11969*.

5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, and 152 others. 2025a. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *Preprint*, arXiv:2508.06471.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, and 150 others. 2025b. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *Preprint*, arXiv:2305.16291.

Yuxuan Wang, Lili Yuan, Karthik Gopalakrishnan, Anusha Narayan-Chen, Alex Fang, and Matthew Hausknecht. 2022. Scienceworld: Is your agent smarter than a 5th grader? In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhenhailong Wang, Haiyang Xu, Junyang Wang, Xi Zhang, Ming Yan, Ji Zhang, Fei Huang, and Heng Ji. 2025. Mobile-agent-e: Self-evolving mobile assistant for complex tasks. *Preprint*, arXiv:2501.11733.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024. Agent workflow memory. In *Forty-second International Conference on Machine Learning*.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. 2025. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *Preprint*, arXiv:2511.20857.

Cheng-Kuang Wu, Zhi Rui Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. 2024. Streambench: Towards benchmarking continuous improvement of language agents. *Preprint*, arXiv:2406.08747.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. 2025. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *Preprint*, arXiv:2510.16079.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. 2025. How memory management impacts llm agents: An empirical study of experience-following behavior. *Preprint*, arXiv:2505.16067.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *arXiv preprint arXiv:2207.01206*.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2024. Evaluating large language models at evaluating instruction following. *Preprint*, arXiv:2310.07641.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. 2025. Memevolve: Meta-evolution of agent memory systems. *Preprint*, arXiv:2512.18746.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. 2026. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *Preprint*, arXiv:2601.03192.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

A Additional Analyses

A.1 Detailed Benchmark Statistics

Table 5 reports the detailed data distribution of EvoMemEval after consistency filtering and sampling balanced by label. Point-wise labels correspond to `useful`, `neutral`, and `harmful`. Pair-wise labels correspond to memory A being preferred, memory B being preferred, and `tie`.

B Evaluator Prompt Templates

Point-wise Prompt. We present the point-wise evaluator prompt template used in our evaluation below.

Template prompt of score evaluation

(System Prompt)
You are a helpful assistant proficient in analyzing reasoning-intensive tasks.
(Instruction)
You are given one task and one retrieved memory snippet. Judge the utility of the memory for solving the task.
(Notice)
Focus on whether the memory helps successful and efficient task solving. Consider correctness gain, reduced reasoning burden, reduced unnecessary solving steps, and reduced inference token overhead. Do not use writing style or length as a direct reason. Be strict and objective.
(Criteria)
Harmful: the memory is misleading, off-topic, or likely to hurt solving quality or efficiency.
Neutral: the memory has limited relevance, is mostly harmless, but provides little concrete help.
Useful: the memory is relevant and actionable, and is likely to improve success or efficiency.
(Desired Output Format)
Return strict JSON with keys `analysis` and `judgment`. Use exactly one of [[HARMFUL]], [[NEUTRAL]], or [[USEFUL]] as the judgment token.
[The Start of Task Name]
{task_name}
[The End of Task Name]
[The Start of Task Question]
{question}
[The End of Task Question]
[The Start of Retrieved Memory]
{memory}
[The End of Retrieved Memory]

Plan then Judge Prompt. We present the prompt used for the planning analysis in Table 2.

Template prompt of Plan then Judge

(System Prompt)
You are a careful and unbiased evaluator of memory quality for LLM task solving. Always follow the rubric and output the required JSON only.
(Instruction)
You are given one task and one retrieved memory snippet. Use a strict step-by-step process to judge memory quality.
(Step-by-step Workflow)
Step 1 (Task Execution Planning): Based on task description and context, infer a concise execution process for finishing the task without using the retrieved memory.
Step 2 (Memory Utility Evaluation): Based on Step 1 plan, evaluate whether the retrieved memory helps, hurts, or has limited effect on solving correctness or efficiency.
(Notice)
Focus on whether this memory is useful for successful and efficient task solving. Usefulness criteria: ability to improve correctness, reduce reasoning burden, reduce unnecessary solving steps, or reduce inference token overhead. Do not use writing style or length as a direct reason. Be strict and objective.
(Criteria)
Harmful: the memory is misleading, off-topic, or likely to hurt solving quality or efficiency.
Neutral: the memory has limited relevance, is mostly harmless, but provides little concrete help.
Useful: the memory is relevant and actionable, likely to improve success and/or efficiency.
(Desired Output Format)
Return strict JSON with keys `step1_task_execution_plan`, `step2_memory_evaluation`, and `judgment`. Use exactly one of [[HARMFUL]], [[NEUTRAL]], or [[USEFUL]] as the judgment token.
[The Start of Task Name]

Benchmark	Point-wise				Pair-wise			
	Useful	Neutral	Harmful	Total	A	B	Tie	Total
ALFWorld	15	15	15	45	20	20	20	60
BabyAI	27	27	27	81	23	23	23	69
ScienceWorld	16	16	16	48	18	18	18	54
WebShop	42	42	42	126	33	33	33	99
Overall	100	100	100	300	94	94	94	282

Table 5: Detailed benchmark statistics after consistency filtering and sampling balanced by label. Point-wise categories are balanced within each benchmark. Pair-wise categories are stratified by the derived comparison label.

{task_name}
[The End of Task Name]
[The Start of Task Question]
{question}
[The End of Task Question]
[The Start of Retrieved Memory]
{memory}
[The End of Retrieved Memory]

Pair-wise Prompt. We present the pair-wise evaluator prompt template used in our evaluation below.

Template prompt of pair comparison

(System Prompt)
You are a helpful assistant proficient in analyzing reasoning-intensive tasks.
(Instruction)
You are given one task and two retrieved memory snippets. Judge which memory is better for solving the task.
(Notice)
The evaluation target is task utility: correctness gain and efficiency gain. Efficiency includes fewer unnecessary reasoning steps and lower inference token overhead. Avoid position bias and do not use length or style alone as the deciding factor.
(Criteria)
Prefer the memory that is more likely to be useful under the order useful > neutral > harmful. If both memories appear equivalent in utility, choose tie.
(Desired Output Format)
Return strict JSON with keys `analysis` and `judgment`. Use exactly one of [[A]], [[B]], or [[C]], where [[A]] means memory A is better, [[B]] means memory B is better, and [[C]] means tie.
[The Start of Task Name]
{task_name}
[The End of Task Name]
[The Start of Task Question]
{question}
[The End of Task Question]
[The Start of Memory A]
{memory_a}
[The End of Memory A]
[The Start of Memory B]
{memory_b}
[The End of Memory B]