# When Do Memories Break? Chain-of-Stage Diagnosis for LLM Memory Systems

Anonymous ACL submission


## Abstract

Memory-augmented Large Language Models frequently fail to maintain long-term context due to opaque evaluation mechanisms that obscure the root causes of failure. To address this, we introduce MemEval, a chain-of-stage diagnostic framework that decomposes the memory pipeline into extraction, update, retrieval, and utilization. Validated on the LoCoMo benchmark, our automated meta-evaluator employs an Ensemble Voting (EV) strategy, achieving 81.62% consistency with human judgment. Our analysis reveals three critical findings: (1) Extraction is the universal bottleneck, causing information loss at the entry point; (2) Retrieval remains brittle in complex reasoning tasks even when memories are correctly stored; and (3) Scaling base model capacity does not guarantee more reliable memory behavior, as higher-capacity backbones may introduce early-stage misalignment that propagates downstream.MemEval provides a necessary diagnostic tool for transparent, stage-aware memory engineering. Code is available at: https://anonymous.4open.science/r/MemEval-8D41.


## 1 Introduction

Large language models excel in short-term tasks but remain unreliable in sustaining long-term, user-centered dialogue. In extended interactions, models frequently forget user preferences, contradict earlier statements, and fail to leverage shared history. While diverse memory architectures—inspired by cognitive science (Wu et al., 2025; Sherwood et al., 2004) and neurobiology (Gutiérrez et al., 2024)—have been proposed to support persistent agents (Park et al., 2023; Jiang et al., 2024), their practical deployment is hindered by a critical bottleneck: the lack of diagnostic evaluation.

Existing benchmarks (He et al., 2024; Maharana et al., 2024) predominantly rely on monolithic, end-to-end metrics (e.g., QA accuracy). By treating memory as a black box, these methods suffer from error entanglement: a failure in the final output could stem from flawed extraction, unsuccessful updates, missed retrieval, or poor utilization of retrieved context. This lack of granularity obscures the causal origins of errors, making it impossible to determine whether a system requires better storage mechanisms or more precise retrieval logic.

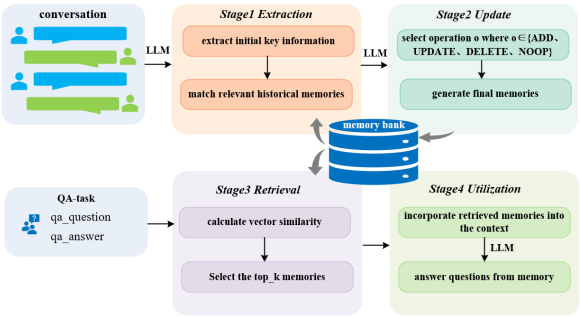


Figure 1: Overview of the Memory Pipeline

We argue that robust memory engineering requires a shift toward probe-based, stage-wise diagnosis. We introduce **MemEval**, a framework adaptable to diverse agentic memory systems that decomposes the memory pipeline into four stages: extraction, update, retrieval, and utilization. By applying MemEval to several memory systems on LoCoMo, we combine expert annotation of 11 error types with an automated meta-evaluator (81.62% human agreement) to pinpoint systemic failures.

In summary, our contributions are:

- A chain-of-stage diagnostic framework for localizing memory failures in LLM systems;
- The design and validation of a scalable, LLM-based meta-evaluator, an automated diagnostic tool that achieves 81.62% consistency with human judgments, enabling efficient, large-scale analysis of memory systems;
- System-level evidence that generation and re-

trieval are the weakest links, and that end-to-end scaling cannot compensate for stage-level misalignment;

- A comprehensive diagnostic study across multiple representative memory architectures;
- A paradigm shift toward causal, pipeline-aware memory engineering, grounded in fine-grained diagnostics.

## 2 Related Work

Large language models increasingly integrate explicit memory mechanisms to support long-dialogue and long-context applications, motivating research on storing, updating, retrieving, and reasoning over long-term information. A broad line of memory-augmented architectures has emerged beyond classical Retrieval-Augmented Generation (RAG).

Systems such as MemGPT (Packer et al., 2024) manage information through OS-like paging; MemoryBank (Zhong et al., 2024), Zep (Rasmussen et al., 2025), and related approaches build persistent, temporally-aware personal knowledge graphs; and recent frameworks including A-Mem (Xu et al., 2025), mem0 (Chhikara et al., 2025), MemoryOS (Kang et al., 2025) MemOS (Li et al., 2025), MemAgent (Yu et al., 2025), and modular designs such as Mirix (Wang and Chen, 2025) explore hierarchical, reflective, or autonomous memory management. Despite architectural diversity, evaluation remains dominated by outcome-level metrics that conflate extraction, update, retrieval, and utilization errors, offering limited transparency into where failures arise. Parallel work investigates cognitive inspirations, system-level organization, and domain-specific memory behaviors. Research on self-organizing memory (Nan et al., 2025), long-context reasoning (Li et al., 2024; Liu et al., 2023), and timeline-based or orchestrated memory management (Ong et al., 2024; Wang et al., 2024) expands the design space. However, these studies similarly lack diagnostics that can disentangle which component—summarization, consolidation, retrieval gating, or utilization—breaks under stress.

Benchmarks for long-term memory have progressed from static profile checks to dynamic interaction assessments. Early datasets such as MSC (Xu et al., 2021), DuLeMon (Xu et al., 2022), and MemoryBank (Zhong et al., 2024) primarily emphasized persona consistency and single-fact recall within limited context windows. Subsequently, benchmarks like LoCoMo (Maharana et al., 2024) and PerLTQA (Du et al., 2024) introduced greater complexity by incorporating multi-hop reasoning and personal-knowledge integration. More recent evaluations, including DialSim (Kim et al., 2024) and LongMemEval (Wu et al., 2024), further advanced the field by adding realistic temporal dynamics and formalizing key capabilities. Despite these advancements, a critical limitation remains: existing benchmarks predominantly assess systems via end-to-end response accuracy. This "black box" approach leaves the causal chain of memory operations opaque, making it difficult to distinguish between failures in retrieval, updating, or reasoning. Our work addresses this gap through stage-wise, interpretable diagnostics.

## 3 Chain of Stage Diagnostic Framework

We propose a provenance-aware, stage-wise diagnostic approach that reframes the question "When do memories break?" by viewing LLM memory not as a black box but as a causal pipeline. Synthesizing our research on mainstream memory architectures, we formalize the memory process into a four-stage paradigm—extraction, update, retrieval, and utilization (see detailed analysis in Appendix A). Our method decomposes the system according to these stages, tracks the full lifecycle of each memory entry, applies a fine-grained error taxonomy grounded in real failures, and employs a dual-track diagnostic protocol combining expert annotation with LLM-based automation to pinpoint where errors arise and how they propagate.

### 3.1 Memory System paradigm

To support deeper analysis of LLM-based memory systems, we introduce a diagnostic paradigm that replaces monolithic, black-box evaluation with a four-stage view of the complete memory lifecycle (Figure 1). This structured decomposition enables granular, causal interpretation and precise localization of failures.

**Extraction Stage** Dialogue is converted into structured, persistent knowledge via an LLM extractor that distills salient facts and preferences. The fidelity of this step bounds all downstream performance.

**Update Stage** New information triggers consolidation, overwriting, deduplication, or pruning of

the memory store, preventing redundancy and drift while maintaining relevance.

**Retrieval Stage** Given a query, the system identifies and ranks relevant information based on semantic alignment, bridging long-term memory with immediate task needs.

**Utilization Stage** Retrieved content is integrated into outputs as the LLM synthesizes evidence and resolves conflicts, determining whether the model can effectively think with memory.

By framing memory as these four interdependent stages, the framework reveals bottlenecks and guides targeted improvements for more robust memory-augmented LLMs.

### 3.2 Chain-Based Provenance Tracking

Our framework employs a lightweight, chain-based provenance tracker that instruments the pipeline, logging full entry lineages via unique IDs and state transitions aligned with interaction turns.

$$\tau(t_i) = \langle q_i,\, a_i^{gold},\, a_i^{pred},\, c_i,\, \{s_j\}_{j=1}^{k},\, \mathcal{M}_i,\, \mathcal{U}_i,\, \mathcal{R}_i \rangle,$$

where:

- $q_i$ is the natural language question;
- $a_i^{gold}$ is the human-annotated ground-truth answer;
- $a_i^{pred}$ is the model-generated response;
- $c_i$ denotes the question category from QA pairs;
- $\{s_j\}_{j=1}^{k}$ are the evidence sentences from the dialogue history that support the correct answer;
- $\mathcal{M}_i = \{m_j^{init}\}_{j=1}^{k}$ is the set of initial memory entries generated from the evidence sentences $s_j$;
- $\mathcal{U}_i = \{u_j^{(1)}, u_j^{(2)}, \ldots, u_j^{(n_j)}\}_{j=1}^{k}$ is the update chain for each memory $m_j^{init}$, recording all revisions over time;
- $\mathcal{R}_i$ is the set of memories retrieved during the processing of $q_i$, including their relevance scores and ranks.

We implement this instrumentation as a lightweight middleware around the system's memory modules, enabling full observability without modifying internal logic. The resulting causal trace reveals whether failures stem from faulty extraction, lossy updates, missed retrieval, or ignored evidence, turning diagnosis from black-box testing into transparent, evidence-driven analysis.

### 3.3 Stage-Wise Error Taxonomy

Building upon the proposed four-stage paradigm, we establish a fine-grained taxonomy comprising 11 distinct memory failure subtypes. To ensure the diagnostic framework is theoretically sound, we model the memory pipeline as a series of functional transformations, where each error subtype is defined as a specific mathematical deviation from the ideal operation. Representative diagnostic examples are visualized in Figure 2.

**Extraction Errors** We model extraction as a mapping function $f : \mathcal{D} \rightarrow \mathcal{M}_{ext}$, aimed at approximating the ideal ground-truth set $\mathcal{M}_{gold}$. According to Set Theory, the deviation relationships between two sets are limited to three mutually exclusive possibilities:

- *1.1 Missing Critical Information*: A proper subset relation ($\mathcal{M}_{ext} \subsetneq \mathcal{M}_{gold}$), indicating essential elements of the ground truth are omitted during extraction.
- *1.2 Incorrect Information*: A difference set violation ($\mathcal{M}_{ext} \setminus \mathcal{M}_{gold} \neq \emptyset$), representing hallucinated elements that emerge in the extraction but are absent in the ground truth.
- *1.3 Ambiguous Information*: A mapping resolution failure. While elements exist within the intersection $\mathcal{M}_{ext} \cap \mathcal{M}_{gold}$, their semantic descriptions fail to form an injective mapping to specific entities, formalized as $\exists m \in (\mathcal{M}_{ext} \cap \mathcal{M}_{gold})$ such that $|\{g \in \mathcal{M}_{gold} \mid m \mapsto g\}| > 1$.

**Update Errors** Governs the transition $\mathcal{T} : S_t \times \Delta \rightarrow S_{t+1}$, where $S_t$ is the memory state and $\Delta$ is new information.

- *2.1 Erroneous updates*: A valid state $m \in S_t$ is wrongly overwritten by an invalid value $m'$, introducing noise into the lifecycle.
- *2.2 Deletion errors*: Persistence failures where a relevant memory is removed without a valid forgetting signal ($m \in S_t \wedge m \notin S_{t+1}$).

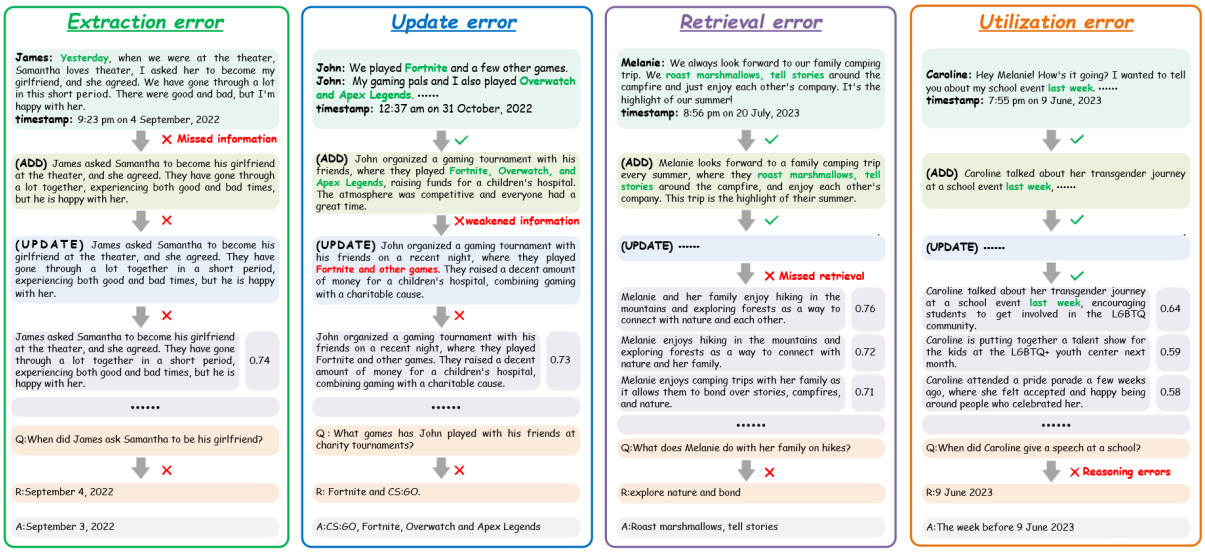


Figure 2: Diagnostic Examples of Memory Failures

- *2.3 Information weakening*: Lossy compression during updates where the information gain decreases, i.e., $I(S_{t+1}) < I(S_t)$.

**Retrieval Errors** We formulate retrieval as a sorting and truncation process $R(q, \mathcal{M}) \rightarrow \mathcal{M}_{topK}$ based on the probability $P(m|q)$. For any ideal memory $m^*$ that should be recalled, failure falls strictly into a binary logical state:

- *3.1 Missed Retrieval*: A failure of set membership ($m^* \notin \mathcal{M}_{topK}$), indicating the relevant memory is entirely excluded from the retrieved subset.

- *3.2 Unreasonable Ranking*: A failure of ordinal position ($m^* \in \mathcal{M}_{topK}$), where the memory is retrieved, but its attention weight is squeezed by noise, causing downstream truncation.

**Utilization Errors** Modeled as a generative function $a = G(q, \mathcal{R}, \theta)$, distinct error states are defined strictly via causal dependency and logical entailment:

- *4.1 Ignoring Evidence*: A causal independence failure ($P(a \mid q, \mathcal{R}, \theta) \approx P(a \mid q, \emptyset, \theta)$). The model bypasses context $\mathcal{R}$ and samples entirely from internal priors $\theta$.

- *4.2 Reasoning Errors*: A logical entailment failure. While causal dependency exists ($P(a \mid q, \mathcal{R}, \theta) \neq P(a \mid q, \emptyset, \theta)$), the mapping from $\mathcal{R}$ to $a$ violates entailment ($\mathcal{R} \nvDash a$).

- *4.3 Format/Detail Errors*: A constraint compliance failure. The answer $a$ violates auxiliary query constraints $\Omega$ despite successful context routing and logical entailment.

### 3.4 Dual-Track Diagnostic Protocol

To validate our taxonomy and analyze how model scale affects memory reliability, we run a dual-track diagnostic study on LoCoMo, where each QA instance pairs a full dialogue history with a retrospective question and the model's final answer.

**Human Annotation Track** A core principle of our manual annotation protocol is strict root-cause attribution to avoid conflating cascading errors. Because failures in early stages propagate downstream, annotators were instructed to identify the earliest point where the system deviated, labeling only that stage as the root cause and marking subsequent stages as "Not Applicable". Using this procedure, expert annotators examined all 1,500 QA instances—each with full dialogue, the final QA pair, and stage-wise provenance—assigning both the root-cause stage and one of 11 error subtypes. Disagreements were resolved through discussion, yielding a high-quality gold-standard dataset.

**Automated Annotation Track** To enable scalable and reliable diagnostics, we introduce MemEval, an automated framework designed to mimic human root-cause analysis. The architecture of this tool was established through a rigorous comparative analysis of three distinct evaluative paradigms.

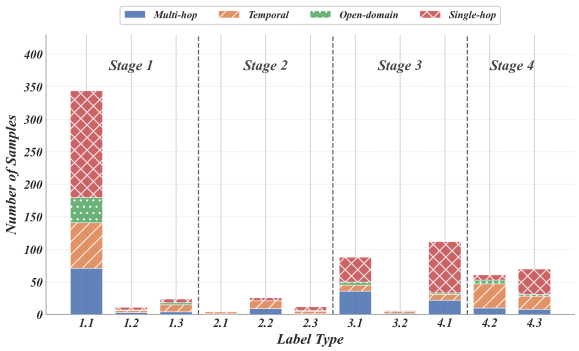


Figure 3: Absolute frequency of human-annotated errors across pipeline stages.

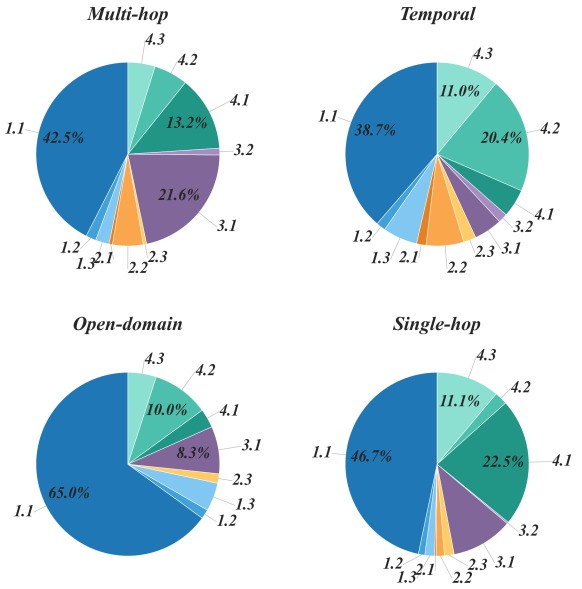


Figure 4: Distribution of error stages by QA category. Note the task-specific failures.

Let $\mathcal{X}$ denote the diagnostic input tuple (comprising context, question, answer, and provenance), and let $\mathcal{E}$ be the set of LLM evaluators available for diagnosis. The goal is to predict the error label $y \in \mathcal{Y}$. We formulate the three paradigms as follows:

***Single-Model Inference*** This paradigm utilizes a solitary evaluator $e \in \mathcal{E}$ with few-shot in-context learning to guide diagnostic reasoning. The prediction is generated directly via a mapping function $\Phi$, representing the prompt-guided inference process as $\hat{y}_{\text{single}} = \Phi(e, \mathcal{X})$, where $\Phi$ encapsulates the reasoning logic defined by the specific instructions.

***Multi-Agent Discussion*** This paradigm employs a committee $\mathcal{E}$ to collaboratively adjudicate each diagnostic stage. If independent judgments fail to reach unanimous consensus, evaluators engage in an iterative debate (up to round $T$) to refine their reasoning based on peer feedback. The final stage-decision is derived via majority voting, formulated as $\hat{y}_{\text{discuss}} = \text{argmax}_{y \in \mathcal{Y}} \sum_{e_i \in \mathcal{E}} \mathbb{I}(y_i^{(T)} = y)$, where instances of complete divergence are resolved by prioritizing the judgment of the designated leading evaluator $e^* \in \mathcal{E}$.

***Ensemble Voting*** This mechanism aggregates independent verdicts from the diverse set of evaluators $\mathcal{E}$ to form a collective decision. Instead of a rigid threshold, the final attribution $\hat{y}_{\text{vote}}$ follows a hierarchical majority-vote protocol: we first calculate the consensus count for each label candidate $l$ as $\sum_{e \in \mathcal{E}} \mathbb{I}(\Phi(e, \mathcal{X}) = l)$. If the maximum consensus exceeds 1, we select the majority label; otherwise, in cases of complete divergence, we defer to the verdict of a designated high-capability evaluator $e_{\text{leader}}$.

In this framework, each model independently receives the full input $\mathcal{X}$ and, guided by structured instructions and few-shot examples (Appendix E), mimics human root-cause analysis by inspecting the memory pipeline stage by stage. Once a failure is identified, the model halts further evaluation and outputs the predicted stage, error subtype, and a brief justification. Final attributions are determined through hierarchical voting across models, reducing single-model variance while preserving the logical rigor of the diagnostic procedure.

## 4 Experiments and Results

Our analysis is centered on four primary research questions:

**RQ1: Localization** At which specific stages of the pipeline do memory failures occur?

**RQ2: Automation** Is it feasible for LLMs to automate diagnostic annotation?

**RQ3: Capacity Impact** How does the capacity of the base model affect memory performance?

**RQ4: Generalization** How robustly do evaluators generalize across diverse memory systems?

### 4.1 Experimental Setup

Our evaluation combines a representative memory system with a benchmark tailored for memory-dependent reasoning.

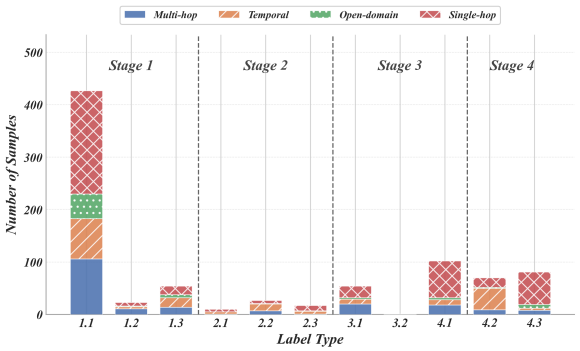


Figure 5: Absolute frequency of LLM-generated diagnostic labels.

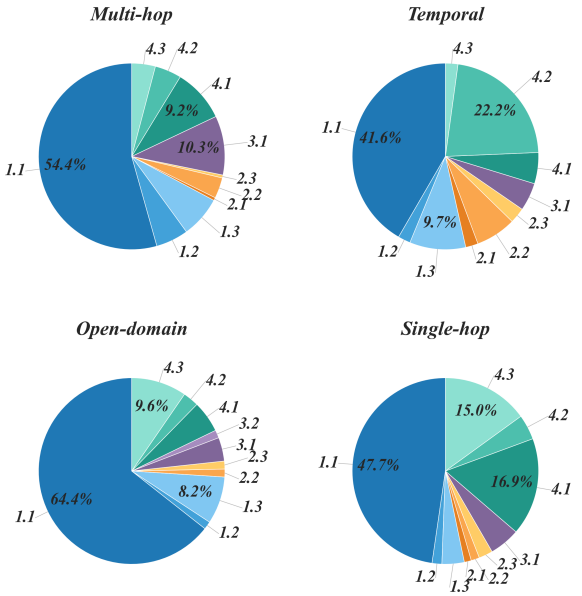


Figure 6: Relative distribution of LLM-generated error labels.

**Memory System** We designate **mem0** as the primary testbed for a comprehensive analysis. To assess the broader applicability of our observations, we complement this with sampled evaluations of **A-mem** and **MemoryOS**.

**Benchmark Dataset** We designate the **LoCoMo** benchmark (comprising over 1,500 dialogue-grounded QA pairs) as our primary diagnostic testbed. Unlike simple recall tasks, LoCoMo stresses distinct pipeline components—Multi-hop questions are retrieval-heavy, whereas Temporal ones expose update and utilization issues. To validate the generalization of our findings across different data distributions, we complement this with a targeted evaluation of 30 sampled questions from the **LongMemEval** benchmark, with detailed results deferred to Appendix C.

## 4.2 Where Do Memories Break? Stage-Level Bottlenecks

We investigate memory breakdown mechanisms by analyzing over 1,500 human-annotated failure cases. Synthesizing the absolute error frequency (Figure 3) with the relative distribution (Figure 4), our analysis reveals a hierarchical failure pattern.

The primary systemic bottleneck is Missing critical information (Label 1.1), which constitutes the global maximum across all tasks, particularly in Open-domain scenarios (65.0%). This initial barrier prevents the encoding of critical information, rendering downstream modules ineffective. Once this barrier is overcome, secondary failures align with specific task complexities: Multi-hop tasks predominantly fail at Missed retrieval (Label 3.1, 21.6%) due to missing semantic links between entities; Temporal tasks falter during Reasoning errors (Label 4.2, 20.4%) when processing chronological logic; and Single-hop tasks exhibit significant Ignoring evidence (Label 4.1, 22.5%), where the model generates outputs that contradict the retrieved evidence.

In summary, memory systems fail primarily at the entry point due to extraction failures, creating a permanent information void; if extraction succeeds, they fail at the specific processing step that imposes the highest cognitive demand for the given task.

## 4.3 Can LLMs Automate Diagnosis? Human vs. Automated Evaluation

We evaluated the fidelity of our diagnostic tool by comparing its generated labels against the human-annotated baseline, analyzing the absolute error frequency followed by the relative distribution.

**General Alignment of Diagnostic Distributions** Applying the Ensemble Voting (EV) strategy, the automated diagnosis reveals a clear hierarchical failure pattern. As illustrated in the absolute frequency analysis (Figure 5), Extraction Failure (Label 1.1) is identified as the universal bottleneck across all categories. This finding demonstrates high consistency with human judgment, particularly in Open-domain and Temporal tasks where error distributions are nearly identical. However, the relative distributions (Figure 6) highlight two key divergences: the tool exhibits higher granularity in Single-hop tasks by explicitly isolating syntactic Format Errors (Label 4.3), whereas in

complex Multi-hop scenarios, it tends to conflate retrieval failures (Label 3.1) with extraction deficits, resulting in an under-reporting of retrieval issues compared to human annotations.

| Strategy | Model | Cat | Stage1 | Stage2 | Stage3 | Stage4 |
|---|---|---|---|---|---|---|
| SMI | gpt-5 | 1 | 94.9 | 36.4 | 34.2 | 52.5 |
| | | 2 | 92.9 | 94.7 | 41.7 | 77.3 |
| | | 3 | 90.7 | 100.0 | 100.0 | 27.3 |
| | | 4 | 94.3 | 75.0 | 48.7 | 74.6 |
| | deepseek-v3.2-exp | 1 | 94.9 | 45.5 | 39.5 | 40.0 |
| | | 2 | 96.4 | 78.9 | 33.3 | 45.5 |
| | | 3 | 97.7 | 100.0 | 40.0 | 18.2 |
| | | 4 | 96.6 | 83.3 | 35.9 | 54.8 |
| | gpt-4.1 | 1 | 88.5 | 54.5 | 15.8 | 35.0 |
| | | 2 | 96.4 | 36.8 | 16.7 | 21.2 |
| | | 3 | 95.3 | 100.0 | 0.0 | 9.1 |
| | | 4 | 89.1 | 58.3 | 28.2 | 69.8 |
| MAD | All | 1 | 94.9 | 54.5 | 36.8 | 47.5 |
| | | 2 | 95.2 | 68.4 | 41.7 | 69.7 |
| | | 3 | 93.0 | 0.0 | 20.0 | 27.3 |
| | | 4 | 94.3 | 75.0 | 46.2 | 68.3 |
| EV | All | 1 | 97.4 | 45.5 | 36.8 | 57.5 |
| | | 2 | 96.4 | 94.7 | 41.7 | 71.2 |
| | | 3 | 95.3 | 100.0 | 60.0 | 27.3 |
| | | 4 | 95.4 | 75.0 | 46.2 | 74.6 |

Table 1: Per-category Stage-level match rates (%) across three diagnostic strategies: **SMI** (Single-Model Inference), **MAD** (Multi-Agent Discussion), and **EV** (Ensemble Voting). **Cat** indices correspond to **1** (Multi-hop), **2** (Temporal), **3** (Open-domain), and **4** (Single-hop).

**Coarse-Grained Comparison of Diagnostic Strategies** As shown in Table 1, comparative analysis reveals that while Extraction (Stage 1) is a solved problem with consistently high fidelity ($> 90\%$), the diagnostic bottleneck lies in the Retrieval and Utilization stages. Weaker single models exhibit catastrophic failures in these complex stages (e.g., 0.0% accuracy in Open-domain retrieval). Notably, Multi-Agent Discussion (MAD) proves insufficient to mitigate these deficits, often underperforming strong baselines due to collective hallucinations. Consequently, Ensemble Voting (EV) establishes itself as the most robust framework, effectively smoothing out individual model variance and achieving superior alignment with human judgment across all categories.

**Fine-Grained Analysis of Ensemble Voting** As shown in the confusion matrix (Figure 7), the tool reliably identifies Stage 1 (96.0%) and Stage 2 (76.7%) boundaries. However, an upstream conflation bias emerges in Stage 3: 40.4% of retrieval failures are misclassified as extraction errors, indicating a tendency to blame initial encoding rather

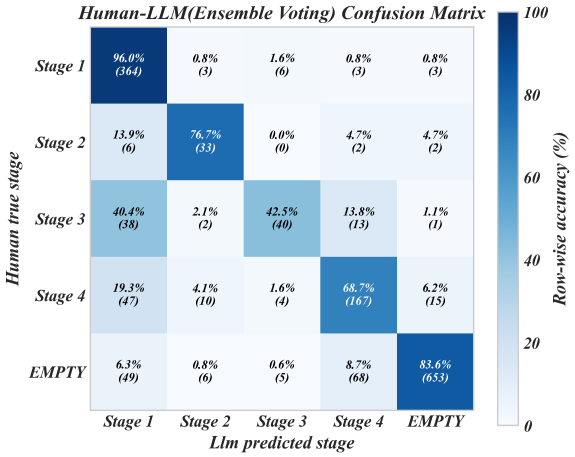


Figure 7: Confusion matrix between human annotations and LLM-generated labels

than the retrieval mechanism.

Despite this bias, the consistency analysis (Table 3) confirms the voting mechanism's robustness for fault localization. Stage Match (81.62%) consistently outperforms strict Exact Match (76.04%). While complex Multi-hop scenarios make distinguishing closely related subtypes difficult, this high overall agreement proves the tool acts as a reliable proxy for locating general failure stages.

Ultimately, while automated evaluation is highly scalable, LLMs still struggle to perfectly trace subtle cascading errors. Human oversight remains essential, highlighting the need for targeted prompting and training to reduce these attribution biases in future evaluators.

### 4.4 Does Stronger mean Better? The Role of Base Model Capacity

To assess the impact of model capacity, we evaluated the Mem0 architecture with GPT-4.1-mini and GPT-4o-mini on the LoCoMo dataset, conducting a detailed analysis grounded in human annotation. As detailed in Table 2, the overall accuracy stagnates at approximately 50% regardless of backbone strength. This empirical evidence reveals that scaling base model capacity alone fails to resolve the structural brittleness of memory systems; instead of improving reliability, it merely shifts the locus of pipeline misalignment.

The more capable GPT-4.1-mini exhibits pronounced misalignment at the earliest stage of the memory pipeline, incurring substantial Extraction errors (408) that propagate into Retrieval failures (228) and hinder effective data access. Qualitative analysis reveals that these errors often arise

| Model | Cat | 1.1 | 1.2 | 1.3 | 2.1 | 2.2 | 2.3 | 3.1 | 3.2 | 4.1 | 4.2 | 4.3 | Right | Total | Labels | ACC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPT-4.1-mini | 1 | 84 | 1 | 0 | 0 | 0 | 1 | 69 | 2 | 7 | 9 | 3 | 113 | 282 | 169 | 40.07 |
| | 2 | 72 | 3 | 1 | 10 | 2 | 1 | 31 | 9 | 2 | 13 | 7 | 167 | 321 | 154 | 52.02 |
| | 3 | 34 | 0 | 0 | 0 | 0 | 0 | 11 | 1 | 2 | 9 | 1 | 39 | 96 | 57 | 40.62 |
| | 4 | 218 | 4 | 0 | 3 | 0 | 4 | 117 | 4 | 13 | 15 | 3 | 457 | 841 | 384 | 54.34 |
| | Total | 408 | 8 | 1 | 13 | 2 | 6 | 228 | 16 | 24 | 46 | 14 | 776 | 1540 | 766 | 50.39 |
| GPT-4o-mini | 1 | 71 | 3 | 4 | 1 | 9 | 1 | 36 | 2 | 22 | 10 | 8 | 167 | 282 | 115 | 40.78 |
| | 2 | 70 | 3 | 11 | 3 | 12 | 4 | 9 | 3 | 9 | 37 | 20 | 181 | 321 | 140 | 43.61 |
| | 3 | 39 | 1 | 3 | 0 | 0 | 1 | 5 | 0 | 2 | 6 | 3 | 60 | 96 | 36 | 37.50 |
| | 4 | 164 | 4 | 6 | 1 | 5 | 6 | 38 | 1 | 79 | 8 | 39 | 351 | 841 | 490 | 58.26 |
| | Total | 344 | 11 | 24 | 5 | 26 | 12 | 88 | 6 | 112 | 61 | 70 | 759 | 1540 | 781 | 50.71 |

Table 2: Combined per-category counts and summary statistics for GPT-4.1-mini and GPT-4o-mini.

| Category | Exact Match (%) | Stage Match (%) |
|---|---|---|
| Multi-hop | 68.44 | 72.34 |
| Temporal | 77.88 | 87.54 |
| Open-domain | 66.67 | 71.88 |
| Single-hop | 78.95 | 83.59 |
| Overall | 76.04 | 81.62 |

Table 3: Consistency between Human and LLM-vote annotations.

| Model | Exact Match (%) | Stage Match (%) |
|---|---|---|
| A-mem | 83.33 | 86.67 |
| MemoryOS | 83.33 | 87.50 |

Table 4: MemEval results for A-mem and MemoryOS on a subset of LoCoMo dialogues.

from missing subjects and confused entity references during memory formation. By contrast, the weaker GPT-4o-mini alleviates many retrieval issues—reducing Label 3.1 errors to 88—but exposes downstream misalignment, with Ignoring Evidence (Label 4.1) errors increasing to 112. Overall, these results show that increasing base model capacity does not improve memory reliability but instead shifts the dominant failure mode, highlighting the need for architectural optimization beyond parameter scaling.

### 4.5 Generalizability of LLM Evaluators Across Memory Architectures

To assess the stability and generalizability of MemEval, we applied the LLM meta-evaluator to A-mem and MemoryOS on a sampled subset of 30 LoCoMo dialogues. As shown in Table 4, the evaluator demonstrates consistent reliability across different architectures. Specifically, it achieves an Exact Match of 83.33% on both systems, with Stage Match scores of 86.67% and 87.50%, respectively. These high agreement rates indicate that our tool is robust in identifying Stage-level failures regardless of the underlying memory implementation. Detailed analyses of the diagnostic results for these systems are provided in Appendix B.

## 5 Conclusion and Future Work

In this work, we present MemEval, a chain-of-stage diagnostic framework that reframes LLM memory evaluation as a transparent, four-stage causal pipeline. Our analysis reveals that memory failures are highly concentrated: Extraction emerges as the primary bottleneck, followed by Retrieval. Crucially, we demonstrate that scaling base model capacity alone fails to resolve these structural vulnerabilities. To enable scalable diagnosis, we validate an LLM-based automated tool that achieves 81.62% stage-level agreement with human experts. By localizing exactly where and how memories break, MemEval facilitates a necessary paradigm shift from black-box testing toward causal, stage-aware memory engineering for persistent AI agents.

Building on these actionable insights, our future work will focus on systematically resolving the key vulnerabilities identified within current architectures. Specifically, we aim to design targeted interventions to mitigate early-stage extraction loss and enhance the robustness of retrieval mechanisms in complex, multi-hop reasoning tasks. Ultimately, we seek to advance the field from precise diagnosis to the development of inherently reliable and resilient long-term memory systems.

## 6 Limitations

While our chain-of-stage framework provides an interpretable approach to diagnosing failures in LLM memory systems, several limitations remain.

First, our LLM-based meta-evaluator, despite strong agreement with human annotations, is not a complete substitute for expert judgment. In complex multi-hop settings, cascading errors can obscure true upstream causes, and the current approach assumes access to detailed provenance traces that may be unavailable in black-box systems.

Second, while MemEval provides a fine-grained, stage-level diagnosis of memory failures, our current scope is strictly diagnostic rather than interventional. We intentionally prioritize establishing a rigorous evaluation baseline over proposing immediate fixes. Consequently, designing stage-specific optimization strategies based on the identified failure modes remains a distinct and critical direction for future research.

Overall, these limitations reflect the inherent challenges of analyzing complex memory pipelines rather than shortcomings of the proposed framework, which we view as a foundational step toward architecture-agnostic memory diagnosis.

## AI-Generated Content Acknowledgement

This paper was assisted in its preparation using generative artificial intelligence tools. Specifically, the Large Language Model (LLM) ChatGPT and Qwen were used to assist with:

- Drafting and iterating on utility scripts for data processing and visualization.
- refining the abstract and conclusion sections for clarity and conciseness.
- Proofreading the manuscript for grammatical errors and stylistic consistency.

The authors are responsible for the final content, including all technical claims, data interpretation, and conclusions presented in this work. All figures, tables, and core experimental results were generated by the authors without direct AI generation.

## References


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

Yiming Du, Hongru Wang, Zhengyi Zhao, Bin Liang, Baojun Wang, Wanjun Zhong, Zezhong Wang, and Kam-Fai Wong. 2024. Perltqa: A personal long-term memory dataset for memory classification, retrieval, and synthesis in question answering. *arXiv preprint arXiv:2402.16288*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *arXiv preprint arXiv:2405.14831*.

Junqing He, Liang Zhu, Rui Wang, Xi Wang, Reza Haffari, and Jiaxing Zhang. 2024. Madial-bench: Towards real-world evaluation of memory-augmented dialogue generation. *arXiv preprint arXiv:2409.15240*.

Xun Jiang, Feng Li, Han Zhao, Jiaying Wang, Jun Shao, Shihao Xu, Shu Zhang, Weiling Chen, Xavier Tang, Yize Chen, and 1 others. 2024. Long term memory: The foundation of ai self-evolution. *arXiv preprint arXiv:2410.15665*.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. *arXiv preprint arXiv:2506.06326*.

Jiho Kim, Woosog Chay, Hyeonji Hwang, Daeun Kyung, Hyunseung Chung, Eunbyeol Cho, Yohan Jo, and Edward Choi. 2024. Dialsim: A real-time simulator for evaluating long-term dialogue understanding of conversational agents. *arXiv e-prints*, pages arXiv–2406.

Yanyang Li, Shuo Liang, Michael R. Lyu, and Liwei Wang. 2024. Making long-context language models better multi-hop reasoners. *arXiv preprint arXiv:2408.03246*.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, and 1 others. 2025. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. 2023. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. *arXiv preprint arXiv:2311.08719*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. 2025. Nemori: Self-organizing agent memory inspired by cognitive science. *arXiv preprint arXiv:2508.03341*.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seungwon Hwang, Dongha Lee, and Jinyoung Yeo.

2024. Towards lifelong dialogue agents via timeline-based memory management. *arXiv preprint arXiv:2406.10996*.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

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

Lauralee Sherwood, Robert Thomas Kell, and Christopher Ward. 2004. *Human physiology: from cells to systems*. Thomson/Brooks/Cole.

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

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

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

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *arXiv preprint arXiv:2504.15965*.

Jing Xu, Arthur Szlam, and Jason Weston. 2021. Beyond goldfish memory: Long-term open-domain conversation. *arXiv preprint arXiv:2107.07567*.

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

Xinchao Xu, Zhibin Gou, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, and Shihang Wang. 2022. Long time no see! open-domain conversation with long-term persona memory. *arXiv preprint arXiv:2203.05797*.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and 1 others. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 19724–19731.

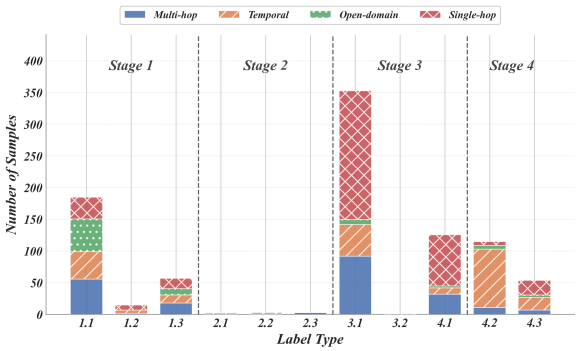


Figure 8: A-mem Stage-Level Error Distribution Across Labels

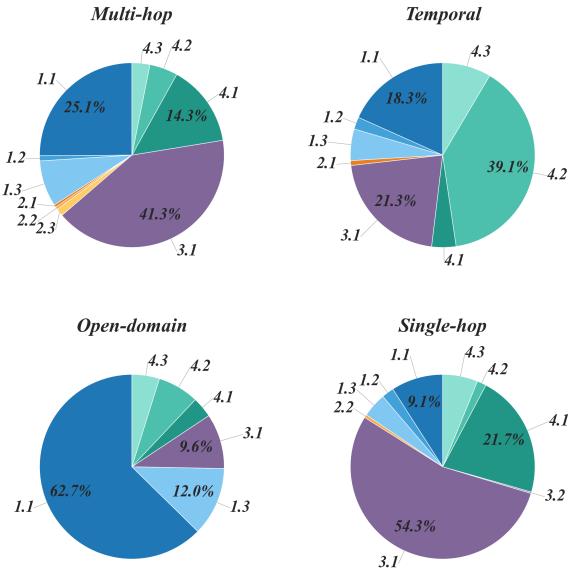


Figure 9: A-mem Error Composition by QA Category

## A The Four-Stage Memory Paradigm

Despite the architectural diversity in current Large Language Model (LLM) agent memory systems, a unified Four-Stage Memory Paradigm can be distilled from analyzing representative state-of-the-art frameworks, namely Mem0, A-mem and MemoryOS. This paradigm, as illustrated in Table 5, delineates the standard lifecycle of long-term memory from unstructured data to effective context:

**Extraction Stage** This stage addresses the challenge of information compression. Systems convert heterogeneous dialogue streams into structured memory units. Mem0 extracts facts and triplets, A-mem constructs atomic notes with contextual descriptions, while MemoryOS adopts a hierarchical approach, extracting dialogue pages, topic segments, and user personas across different storage tiers.

**Update Stage** This stage maintains the consistency and currency of the knowledge base. Mem0 employs tool-use mechanisms for CRUD operations; A-mem emphasizes memory evolution through link generation and content refinement; MemoryOS introduces operating system principles, utilizing heat-based algorithms and FIFO mechanisms to migrate, promote, or evict data between memory tiers.

**Retrieval Stage** The goal here is to locate high-fidelity context within the memory store. Approaches range from Mem0's vector search to A-mem's associative linking. MemoryOS distinguishes itself with a multi-tiered strategy, specifically employing a two-stage (segment-to-page) retrieval process in its mid-term memory to balance scope and precision.

**Utilization Stage** Finally, the retrieved structured memories are integrated into the generation process. Whether injecting denoised facts (Mem0), graph-aided context (A-mem), or a hierarchy of recent, historical, and personalized information (MemoryOS), this stage ensures that the LLM's generation is grounded in a coherent and adaptive long-term context. This paradigm provides a comprehensive theoretical framework for evaluating and designing future agentic memory systems, bridging the gap between vector-based, note-based, and OS-inspired architectures.

## B Application of MemEval on other Memory Systems

### B.1 A-mem

The diagnostic results for A-mem, visualized in Figures 8 and 9, reveal a distinct, category-dependent failure profile that highlights specific architectural bottlenecks. For Single-hop and Multi-hop queries, the dominant failure mode is Missed Retrieval (Label 3.1). As shown in the line chart (Figure 8), Single-hop queries exhibit a dramatic spike at label 3.1, which corresponds to the substantial pink sector (54.3%) in the pie charts, indicating that while A-mem may store the information, its retrieval mechanism frequently fails to surface relevant memories due to insufficient semantic matching or strict thresholding. In contrast, Open-domain interactions show a completely different error distribution dominated by Extraction Failures, specifically Missing Critical Information (Label 1.1). The

| Stage | MEM0 | A-MEM | MEMORYOS |
|---|---|---|---|
| **Extraction** | Facts and Triplets | Atomic Notes | Hierarchical Segments & Persona |
| **Update** | CRUD Operations | Link Generation & Evolution | Heat-based Migration & Eviction |
| **Retrieval** | Vector Search | Associative Linking Search | Multi-tiered Hybrid Retrieval |
| **Utilization** | Denoised Prompt Injection | Graph-aided Prompting | Hierarchical Context Integration |

Table 5: Comparative Analysis of Representative Memory Architectures. We summarize the distinct approaches of Mem0, A-mem, and MemoryOS across the proposed four-stage paradigm.

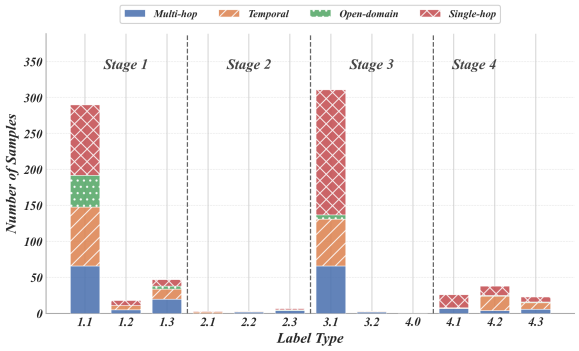


Figure 10: MemoryOS Stage-Level Error Distribution Across Labels

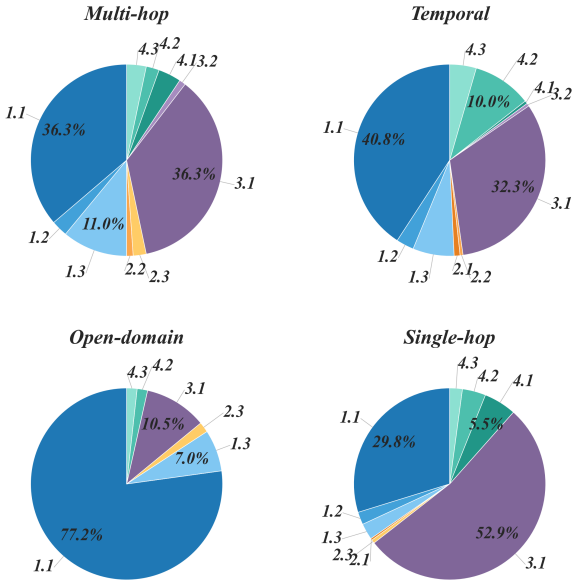


Figure 11: MemoryOS Error Composition by QA Category

pie chart shows that 62.7% of errors in this category stem from the extraction Stage, suggesting that A-mem's extractor struggles to capture subtle, unstructured details typical of casual conversation. Furthermore, for Temporal queries, the diagnostic shifts towards the Utilization stage with a significant spike at Reasoning Errors (Label 4.2) (39.1%), revealing that even when A-mem successfully retrieves time-related facts, the underlying LLM frequently fails to perform the necessary chronological reasoning. Overall, MemEval successfully disentangles these failures, demonstrating that A-mem's weaknesses are highly context-dependent—suffering from recall issues in factual tasks, encoding loss in open chat, and reasoning gaps in temporal scenarios.

## B.2 MemoryOS

The diagnostic landscape for MemoryOS, depicted in Figures 10 and 11, exhibits a pervasive tendency towards Extraction Failures (Label 1.1) across all categories, contrasting sharply with the stage-specific bottlenecks observed in A-mem. This extraction deficit is most pronounced in Open-domain dialogues, where a staggering 77.2% of failures stem from the system's inability to capture critical information, indicating an overly aggressive filtering mechanism in its memory encoder. For Single-hop queries, while Missed Retrieval (Label 3.1) remains the primary bottleneck (52.9%), a significant portion of errors (29.8%) is still improved by extraction failures, further confirming that information loss occurs early in the pipeline. In Multi-hop scenarios, the system shows a balanced failure distribution between Extraction (36.3%) and Retrieval (36.3%), suggesting a compounded weakness where complex chains are broken either by missing links or recall failures. Notably, in Temporal tasks, unlike A-mem which primarily failed at the reasoning stage, MemoryOS struggles predominantly with Extraction (40.8%) and Retrieval (32.3%). This indicates that for MemoryOS, time-sensitive information is often lost or inaccessible before the utilization stage is even reached, effectively preventing the model from attempting the necessary chronological reasoning.

You are an evaluation assistant. Determine whether qa_response is semantically consistent with qa_answer.

**Consistency rules:**
- All key information in qa_answer must appear in qa_response.
- Missing, incorrect or unclear details make it inconsistent.

**Example :**
qa_answer: "first weekend of August 2023"
qa_response: "5 August 2023."
→ inconsistent (incorrectly narrows the time range)

**Now evaluate:**
**input:**
- qa_question: {qa_question_str}
- qa_answer: {qa_answer_str}
- qa_response: {qa_response_str}

**Output**:
{{
"is_consistent": true/false,
"reason": "brief explanation"
}}

Figure 12: Prompt template for response-level judgment, used to determine whether the model's answer satisfies the question given available memories.

# C Evaluation on the LongMemEval Benchmark

To address potential concerns regarding the generalizability of our findings beyond the LoCoMo benchmark, we extend our evaluation to the LongMemEval dataset.

We randomly sampled $N = 30$ representative long-term QA instances from the `longmemeval_s` subset. Using mem0 (powered by GPT-4o-mini) as the target system, we directly applied MemEval for automated diagnosis.

MemEval successfully parsed all interaction logs without requiring any prompt adaptations. The mem0 system achieved an overall accuracy of 76.7% (23/30). Crucially, consistent with our primary observations on LoCoMo, errors were predominantly concentrated in Stage 1 (Memory Extraction), accounting for 57% (4/7) of all failures. The remaining errors were evenly distributed across the Update, Retrieval, and Utilization stages (14% each).

This seamless adaptation confirms that our error taxonomy and the MemEval pipeline are not overfitted to a specific data distribution. Instead, they demonstrate robust external validity and direct transferability to other long-term memory benchmarks.

| Metric | Value |
|---|---|
| API Calls | ∼1.95 / query |
| Token Cost | ∼2.3k / query |
| Avg. Latency | 11.37s / query |

Table 6: Resource consumption metrics for MemEval under the Single-Model Inference (SMI) configuration.

# D Cost Analysis and Low-Resource Configuration

To address potential concerns regarding computational overhead, we benchmarked MemEval under a low-cost Single-Model Inference (SMI) configuration across the entire LoCoMo dataset (1,982 queries).

As an offline diagnostic framework, real-time execution is not a strict prerequisite; nevertheless, MemEval demonstrates exceptional operational efficiency. As detailed in Table 6, the pipeline consumes only ∼2.3k tokens and requires fewer than two API calls per query. Furthermore, the architecture natively supports full parallel batch processing. With an average latency of 11.37 seconds per instance, the framework introduces a minimal resource footprint, ensuring highly scalable and economical automated evaluations for long-term memory systems.

# E Prompts for MemEval

## E.1 Prompt for Response Judgment

Figure 12 presents the specific prompt template used for response-level judgment. This prompt is designed to evaluate whether the generated response provides sufficient information to correctly address the user's question.

## E.2 Prompt for Extraction Stage

Figure 13 illustrates the prompt designed for the extraction stage. It serves to assess the quality of the initial memories retrieved by the memory system, specifically identifying whether the extracted content is problematic and classifying the error into a specific category.

## E.3 Prompt for Update Stage

Figure 14 shows the prompt employed in the Update stage. This prompt is used to detect potential anomalies during the memory modification process and to classify the specific category of the error.

### E.4 Prompt for Retrieval Stage

In our experimental setup, we set the retrieval parameter top-$k = 10$, retrieving ten memory entries for both Speaker A and Speaker B. Figure 15 illustrates the prompt used to diagnose potential failures in the retrieval stage and to identify the specific error category.

### E.5 Prompt for Utilization Stage

This prompt is employed specifically when the Extraction, Update, and Retrieval stages are verified to be correct. Figure 16 demonstrates the prompt used to assess whether the Large Language Model (LLM) fails to utilize these memories effectively during response generation, and to classify the specific error category.

You are an evaluation assistant for the Memory Extraction Stage.
**Task:**
1. Use their initial_results (and time_stamp if needed) to determine whether the extracted memories are sufficient to answer qa_question.
2. If sufficient → is_sufficient = true (label = null).
3. If insufficient, classify the issue:
   - "1.1": Missing key information
   - "1.2": Incorrect or conflicting information
   - "1.3": Ambiguous or overly generic information

**Examples:**
Example 1:
qa_question: "Where did Caroline move from 4 years ago?"
qa_answer: "Sweden"
qa_response: "home country"
person1_memories: {{"initial_results": ["Caroline moved from her home country 4 years ago"]}}
person2_memories: {{"initial_results": []}}
Output:
{{
  "is_sufficient": false,
  "label": "1.1",
  "reason": "The extracted memory only says 'home country' and lacks the specific detail 'Sweden.'"
}}

**……**

**Now evaluate:**

**Input:**
- qa_question: {qa_question_str}
- qa_answer: {qa_answer_str}
- qa_response: {qa_response_str}
- person1_memories: {memories1_str}
- person2_memories: {memories2_str}

**Output format:**
{{
  "is_sufficient": true/false,
  "label": "1.1" or "1.2" or "1.3" or null,
  "reason": "Detailed explanation"
}}

Figure 13: Prompt template for diagnosing errors in the Extraction stage, focusing on missing, incorrect, or overly generic memory entries.

You are an evaluation assistant for the Memory Update Stage.
**Task:**
1. From the update_chain, use only the final updated memory for each item.
2. Determine whether the updated memories contain sufficient and correct information to answer qa_question.
3. If sufficient → is_sufficient = true (label = null).
4. If insufficient, classify the issue according to the update error type:
   - "2.1": Incorrect update (added wrong or fabricated details)
   - "2.2": Deleted information (removed necessary memory entries)
   - "2.3": Weakened information (kept but diluted or less specific)

**Examples:**
Example 1:
qa_question: "What did James prepare for the first time in the cooking class?"
qa_answer: "Omelette"
qa_response: "omelette, meringue, dough"
update_chain: [{{
  "event": "UPDATE",
  "memory": "James ... made an omelette ... He also made meringue and learned how to make dough.",
  "previous_memory": "James ... made a great omelette for the first time."
}}]
Output:
{{
  "is_sufficient": false,
  "label": "2.1",
  "reason": "The update introduces incorrect new first-time dishes—meringue and dough—contradicting the original memory."
}}

**…**

**Now evaluate:**

**Input:**
- qa_question: {qa_question_str}
- qa_answer: {qa_answer_str}
- qa_response: {qa_response_str}
- person1_memories: {memories1_str}
- person2_memories: {memories2_str}

**Output format:**
{{
  "is_sufficient": true/false,
  "label": "2.1" or "2.2" or "2.3" or null,
  "reason": "Detailed explanation"
}}

Figure 14: Prompt template for the Update stage, identifying incorrect edits, deletions, or contradictions introduced during memory updating.

You are an evaluation assistant for the Memory Retrieval Stage.
**Task:**
Based strictly on speaker1_retrieval and speaker2_retrieval:
1. Determine whether the retrieved memories contain enough correct information to answer qa_question.
2. If sufficient → is_sufficient = true (label = null).
3. If insufficient, determine the retrieval issue:
   - "3.1": Failed to recall correct information (missing the key facts)
   - "3.2": Unreasonable ranking (recalled irrelevant/common info while missing the most relevant facts)

**Examples:**
Example 1:
qa_question: "How does Melanie prioritize self-care?"
qa_answer: "by carving out some me-time each day for activities like running, reading, or playing the violin"
qa_response: "Running, pottery, charity races."
speaker1_retrieval: [
  "Melanie prioritizes her mental health...",
  "Melanie enjoys running as a way to de-stress...",
  "Melanie is thankful for her family..."
]
speaker2_retrieval: [
  "Melanie finds self-care to be a work in progress...",
  "Melanie has been running longer...",
  "Melanie values mental health..."
]
Output:
{{
  "is_sufficient": false,
  "label": "3.1",
  "reason": "The retrieved memories mention running and mental-health efforts but miss key self-care details such as reading, violin, and daily me-time."
}}

**Now evaluate:**
Input:
- qa_question: {qa_question_str}
- qa_answer: {qa_answer_str}
- speaker1_retrieval: {speaker1_memories_str}
- speaker2_retrieval: {speaker2_memories_str}

Output format:
{{
  "is_sufficient": true/false,
  "label": "3.1" or "3.2" or null,
  "reason": "Detailed explanation"
}}

Figure 15: Prompt template for Retrieval-stage evaluation, assessing whether relevant memories are successfully recalled given the query.

You are an evaluation assistant for the Reasoning Stage.

**Context:**
All previous stages (extraction, update, retrieval) have passed, meaning the model had sufficient correct information.
If qa_response still does not match qa_answer, the issue is a reasoning error.

**Task:**
Based on qa_question, qa_answer, qa_response, and the retrieved memories, classify the reasoning issue:
- "4.1": Correct memory entries were ignored (model overlooks correct memory entries present in retrieval)
- "4.2": Reasoning error (model invents details, over-specifies, or makes unsupported inferences)
- "4.3": Format or detail error (minor deviations such as missing qualifiers or altered phrasing that slightly change meaning)

**Examples:**

Example 1:
qa_question: "What does Melanie do with her family on hikes?"
qa_answer: "Roast marshmallows, tell stories"
qa_response: "explore nature and bond"
speaker1_retrieval: ["Melanie prioritizes her mental health..."]
speaker2_retrieval: ["Melanie ... roasted marshmallows ... and told stories..."]
Output:
{{
  "label": "4.1",
  "reason": "The retrieved memory clearly includes roasting marshmallows and telling stories, but the model ignored this memory entry"
}}

**…**
**Now evaluate:**

Input:
- qa_question: {qa_question_str}
- qa_answer: {qa_answer_str}
- qa_response: {qa_response_str}
- speaker1_retrieval: {speaker1_memories_str}
- speaker2_retrieval: {speaker2_memories_str}

Output format:
{{
  "label": "4.1" or "4.2" or "4.3",
  "reason": "Detailed explanation"
}}

Figure 16: Prompt template for the Utilization stage, diagnosing failures in reasoning or response construction despite correct memory retrieval.