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.

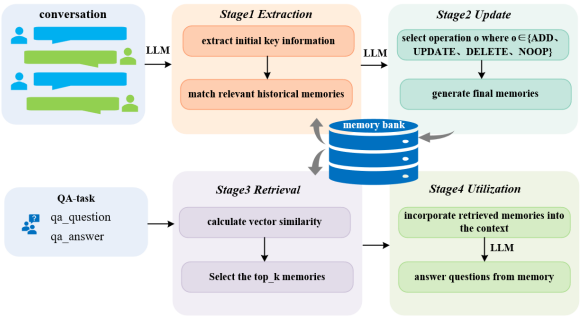

Figure 1: Overview of the Memory Pipeline

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 extracting, 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 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 New information triggers consolidation, overwriting, deduplication, or pruning of the memory store, preventing redundancy and drift while maintaining relevance.

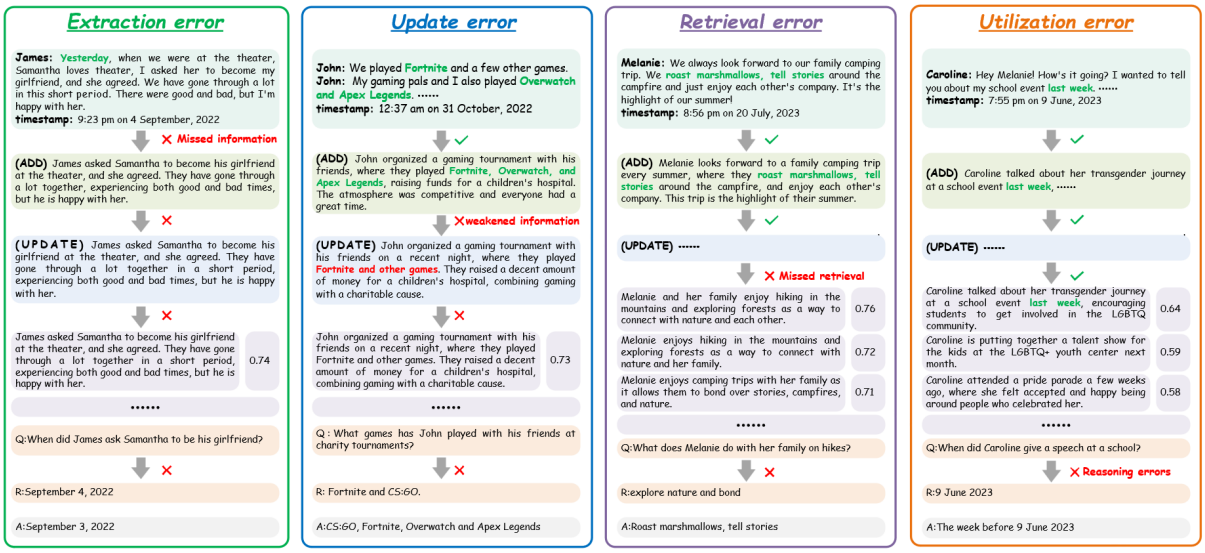

Figure 2: Diagnostic Examples of Memory Failures

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

Utilization 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 is the question category from QA pairs;
- $\{s_j\}_{j=1}^{k}$ are the evidence sentences from the dialogue that support the correct answer;
- $\mathcal{M}_i = \{m_j^{init}\}_{j=1}^{k}$ denotes the initial memories derived from 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

As illustrated by the empirical case analysis in Figure 2, memory systems exhibit distinct failure modes across different functional stages—for instance, as seen in the Extraction Error case, omitting the critical temporal anchor "yesterday" from a dialogue timestamped September 4th. This early extraction failure irrecoverably drops the relative time context, causing the system to incorrectly output the dialogue's timestamp (September 4th) instead of the factual event date (September 3rd) during the downstream QA stage. A detailed discussion of representative errors across all stages, along with an in-depth analysis of their underlying root causes, is provided in Appendix E.

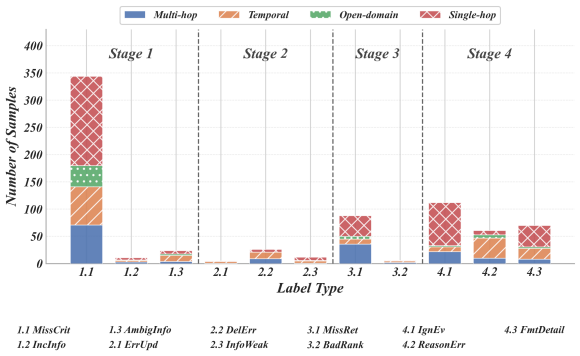

Figure 3: Absolute frequency of human-annotated errors across pipeline stages.

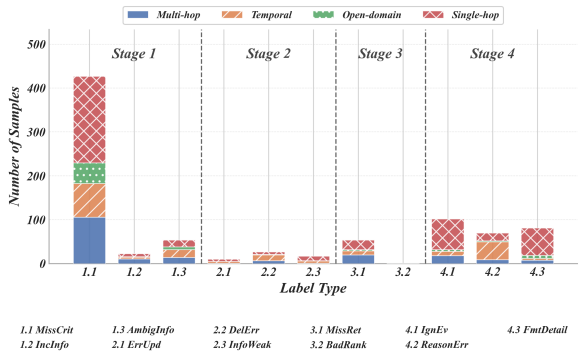

Figure 4: Absolute frequency of LLM-generated diagnostic labels across pipeline stages.

Motivated by these observations and building upon the provenance tracker $\tau(t_i)$, we formalize the memory pipeline as a series of functional transformations over the defined variables. Consequently, we establish a comprehensive and orthogonal taxonomy comprising 11 distinct failure subtypes, where each error represents a specific mathematical deviation from the ideal operation.

Ext. Extraction maps evidence sentences $\{s_j\}_{j=1}^{k}$ to the initial memory set $\mathcal{M}_i$. Deviations from the ideal $\mathcal{M}_i^{gold}$ are categorized as:
1.1 Missing Critical Information: Proper subset ($\mathcal{M}_i \subsetneq \mathcal{M}_i^{gold}$), indicating essential elements of the ground truth are omitted during extraction.
1.2 Incorrect Information: A difference set violation ($\mathcal{M}_i \setminus \mathcal{M}_i^{gold} \neq \emptyset$), representing hallucinated elements that emerge in the extraction but are absent in the ground truth.
1.3 Ambiguous Information: A semantic resolution failure. A memory $m \in \mathcal{M}_i$ lacks necessary granularity, mapping to multiple distinct entities in the ground-truth context and preventing downstream disambiguation.

Upd. Update operations govern the state transitions recorded within the update chain $\mathcal{U}_i$. For any memory transitioning from state $u_j^{(t)}$ to $u_j^{(t+1)}$ upon receiving new dialogue turns:
2.1 Erroneous updates: A valid memory state $u_j^{(t)}$ is wrongly overwritten by an invalid or contradictory value in $u_j^{(t+1)}$, introducing noise.
2.2 Deletion errors: A persistence failure where a relevant memory is dropped from the chain ($u_j^{(t)}$ exists, but is erroneously nullified in $u_j^{(t+1)}$) without a valid forgetting signal.
2.3 Information weakening: A lossy compression during updates where the specific information gain decreases, formalized as $I(u_j^{(t+1)}) < I(u_j^{(t)})$.

Ret. We formulate retrieval as a selection and ranking process that filters the terminal memory pool (the final states within $\mathcal{U}_i$) to populate the retrieved subset $\mathcal{R}_i$. For any ideal critical memory m^* required to answer q_i:
3.1 Missed Retrieval: A failure of set membership ($m^* \notin \mathcal{R}_i$), indicating the relevant memory is entirely excluded from the retrieved context.
3.2 Unreasonable Ranking: A failure of ordinal position ($m^* \in \mathcal{R}_i$), where the memory is retrieved, but its attention weight is squeezed by noise, causing practical truncation before generation.

Util. Modeled as a generative function $a_i^{pred} = G(q_i, \mathcal{R}_i, \theta)$ parameterized by model weights θ. Distinct error states are defined via causal dependency and logical entailment:
4.1 Ignoring Evidence: A causal independence failure ($P(a_i^{pred} \mid q_i, \mathcal{R}_i, \theta) \approx P(a_i^{pred} \mid q_i, \emptyset, \theta)$). Although internal probabilities are inaccessible in black-box LLMs, our framework operationalizes this by prompting the evaluator to detect if a_i^{pred} hallucinated facts derived entirely from prior weights θ while bypassing the provided context $\mathcal{R}_i$.
4.2 Reasoning Errors: A logical entailment failure. While causal dependency on the context exists, the mapping from $\mathcal{R}_i$ to a_i^{pred} violates logical entailment ($\mathcal{R}_i \nvDash a_i^{pred}$).
4.3 Format/Detail Errors: A constraint compliance failure. The predicted answer a_i^{pred} violates auxiliary query constraints Ω 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

Strategy	Model	QA.Cat	Stage1	Stage2	Stage3	Stage4
SMI	gpt-5	A	94.9	36.4	34.2	52.5
		B	92.9	94.7	41.7	77.3
		C	90.7	100.0	100.0	27.3
		D	94.3	75.0	48.7	74.6
	deepseek-v3.2-exp	A	94.9	45.5	39.5	40.0
		B	96.4	78.9	33.3	45.5
		C	97.7	100.0	40.0	18.2
		D	96.6	83.3	35.9	54.8
	gpt-4.1	A	88.5	54.5	15.8	35.0
		B	96.4	36.8	16.7	21.2
		C	95.3	100.0	0.0	9.1
		D	89.1	58.3	28.2	69.8
MAD	All	A	94.9	54.5	36.8	47.5
		B	95.2	68.4	41.7	69.7
		C	93.0	0.0	20.0	27.3
		D	94.3	75.0	46.2	68.3
EV	All	A	97.4	45.5	36.8	57.5
		B	96.4	94.7	41.7	71.2
		C	95.3	100.0	60.0	27.3
		D	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). The **QA.Cat** mapping is as follows: A: Multi-hop, B: Temporal, C: Open-domain, and D: Single-hop.

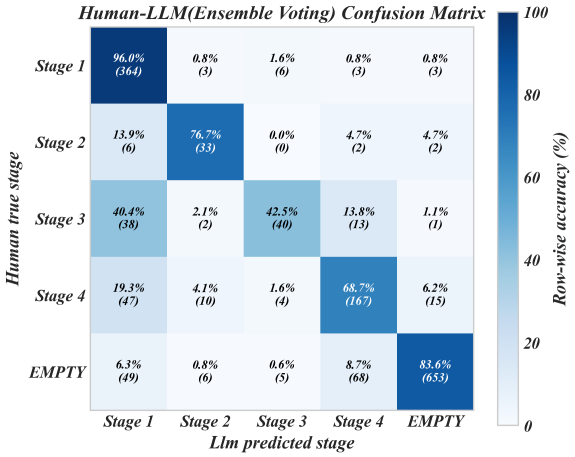

Figure 5: Confusion matrix between human annotations and LLM-generated labels.

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.A detailed analysis of the associated human annotation costs is provided in Appendix F.

Automated Annotation Track To enable scalable 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.

Let $\mathcal{X}$ be the diagnostic input tuple (context, question, answer, provenance) and $\mathcal{E}$ the set of LLM evaluators. To predict the error label $y \in \mathcal{Y}$, we formulate three paradigms:

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 via a mapping function Φ, representing the prompt-guided inference process as $\hat{y}_{\text{single}} = \Phi(e, \mathcal{X})$, where Φ 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 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 calculate the consensus count for each 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_{leader}.

In this framework, each model independently receives the full input $\mathcal{X}$ and, guided by structured instructions and few-shot examples (Appendix I), 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. A detailed analysis of the computational cost and latency, demonstrating the framework's operational efficiency, is provided in Appendix G.

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.

Memory System We use **mem0** (Chhikara et al., 2025) as the primary testbed for our comprehensive analysis. To examine the broader applicability of our observations, we additionally evaluate **A-mem** (Xu et al., 2025), **MemoryOS** (Kang et al., 2025), and the **OpenClaw** framework.

Benchmark We designate the **LoCoMo** (Maharana et al., 2024) 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 generalization of our findings across different data distributions, we complement this with an analysis of 500 questions from the **LongMemEval** (Wu et al., 2024) benchmark.

4.2 Where Do Memories Break? Stage-Level Bottlenecks

We investigate memory breakdown mechanisms by analyzing over 1,500 human-annotated failure cases derived from the mem0 framework powered by the GPT-4o-mini backbone. By examining the absolute error frequency (Figure 3), 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. 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) due to missing semantic links between entities; Temporal tasks falter during Reasoning errors (Label 4.2) when processing chronological logic; and Single-hop tasks exhibit significant Ignoring evidence (Label 4.1), 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

To evaluate the fidelity of our diagnostic tool, we compared its generated labels against the human-annotated baseline across three distinct diagnostic strategies. While Single-Model Inference (SMI) evaluates each backbone (GPT-5, DeepSeek V3.2 Exp, GPT-4.1) individually, Multi-Agent Discussion (MAD) and Ensemble Voting (EV) collaboratively synthesize judgments from all three. We subsequently analyze their absolute and relative error distributions.

General Alignment of Diagnostic Distributions Under the EV strategy, the automated diagnosis demonstrates an overall distribution trend that is highly consistent with human judgment. Most notably, it successfully identifies Extraction Failure (Label 1.1) as the universal bottleneck across all task types (Figure 4), accurately reflecting the macro-level failure patterns. While an examination of the absolute error frequencies reveals minor variances in fine-grained error attribution within specific stages—such as slight shifts between retrieval and extraction deficits in multi-hop scenarios, or stricter format evaluations in single-hop tasks—the primary conclusions regarding stage-level bottlenecks remain robust and well-aligned.

Coarse-Grained Comparison of Diagnostic Strategies As shown in Table 1, while Stage

Model	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 (%)
Closed-source Models															
GPT-4.1-mini	408	8	1	13	2	6	228	16	24	46	14	776	1540	766	50.39
GPT-4o-mini	344	11	24	5	26	12	88	6	112	61	70	759	1540	781	50.71
Open-source Models															
Qwen3.5-35B	289	17	9	9	0	5	53	0	34	40	35	884	1540	656	57.40
Qwen3.5-122B	264	20	12	14	6	11	53	0	33	40	45	898	1540	642	58.31

Table 2: Overall diagnostic results on the LoCoMo benchmark. The labels map to our diagnostic taxonomy: **Stage 1 (Ext.)** 1.1 Miss, 1.2 Incorrect, 1.3 Ambiguous; **Stage 2 (Upd.)** 2.1 Err. Update, 2.2 Delete, 2.3 Weaken; **Stage 3 (Ret.)** 3.1 Miss, 3.2 Rank; **Stage 4 (Util.)** 4.1 Ignore, 4.2 Reason, 4.3 Format.

1 diagnosis maintains consistently high fidelity ($> 90\%$), Retrieval and Utilization remain critical bottlenecks where weaker single models fail catastrophically (e.g., 0.0% accuracy in Open-domain retrieval). Notably, Multi-Agent Discussion (MAD) fails to mitigate these issues, often underperforming strong baselines due to collective hallucinations. Consequently, Ensemble Voting (EV) emerges as the most robust framework, effectively smoothing individual model variance to achieve superior alignment with humain robust and well-alignedjudgment.

Fine-Grained Analysis of Ensemble Voting As shown in Figure 5, the EV strategy reliably isolates Stage 1 (96.0%) and Stage 2 (76.7%) errors, although it exhibits an upstream bias by misclassifying 40.4% of Stage 3 retrieval failures as early-stage extraction deficits. Despite this tendency to attribute failures to initial encoding rather than retrieval, our consistency analysis (detailed in Appendix B) confirms the framework's overall robustness in fault localization. Notably, the Stage Match accuracy (81.62%) consistently outperforms the strict Exact Match baseline (76.04%). Ultimately, while the automated tool serves as a highly reliable proxy for identifying general failure stages, the inherent difficulty LLMs face in tracing subtle cascading errors—particularly in complex Multi-hop scenarios—indicates that targeted human oversight remains necessary to mitigate attribution biases.

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

To assess the impact of base model capacity, we evaluate the Mem0 architecture on LoCoMo across closed-source (GPT-4.1-mini, GPT-4o-mini) and open-source (Qwen3.5-35B, Qwen3.5-122B) families (Table 2). Consistent trends reveal that scaling capacity yields only marginal accuracy improvements. For closed-source models, overall accuracy stagnates at approximately 50%. Rather than resolving the structural brittleness of memory systems, the stronger GPT-4o-mini merely shifts the locus of the bottleneck: while it partially mitigates the severe early-stage Extraction (Label 1.1) and Retrieval (Label 3.1) failures seen in GPT-4.1-mini, it exhibits a significant surge in downstream Utilization errors, such as Ignoring Evidence (Label 4.1). A similar bottleneck is observed in the open-source Qwen family, where scaling from 35B to 122B yields a negligible performance gain (57.40% to 58.31%), and Extraction failures persistently act as the dominant barrier. Ultimately, switching to a stronger backbone shifts failure distributions across functional stages rather than eliminating them, emphasizing the critical need for fundamental architectural optimization over mere capability scaling. Detailed task-specific breakdowns and analyses for both model families are provided in Appendix H.

4.5 Generalizability of MemEval

To assess the stability and universality of our diagnostic framework, we evaluated MemEval along two critical dimensions: cross-architecture applicability and cross-dataset robustness.

Generalization Across Memory Architectures We investigated whether MemEval is agnostic to the underlying system design by applying it to three diverse architectures: A-mem, MemoryOS, and the emerging OpenClaw framework. As detailed in Table 4, evaluation on a sampled subset of LoCoMo dialogues yielded an Exact Match of 83.33% for both A-mem and MemoryOS, alongside high Stage Match scores (86.67% and 87.50%, respectively). Furthermore, a full-scale evaluation of OpenClaw (Table 3) reveals that while it achieves a high end-to-end accuracy of 73.70%, MemEval successfully isolated 405 granular errors without re-

QA.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 (%)
A	23	3	6	1	2	2	32	0	2	29	3	179	282	103	63.48
B	19	1	1	4	0	3	24	0	31	6	12	228	321	93	71.03
C	33	2	0	0	0	0	5	0	0	3	2	51	96	45	53.12
D	44	7	3	2	1	8	54	0	1	32	9	677	841	164	80.50
Total	119	13	10	7	3	13	115	0	34	70	26	1135	1540	405	73.70

Table 3: Diagnostic results of the OpenClaw architecture on the LoCoMo benchmark. The **QA.Cat** mapping is A: Multi-hop, B: Temporal, C: Open-domain, D: Single-hop.

quiring alterations to the diagnostic taxonomy. This demonstrates that MemEval effectively reveals localized, stage-specific flaws despite high overall performance metrics, thereby providing valuable guidance for the optimization of emerging memory architectures. Detailed evaluation results and comprehensive analyses for all three memory systems are provided in Appendix C.

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.

Generalization Across Datasets Beyond architectural flexibility, we extended our evaluation to the entire LongMemEval-S benchmark (500 representative long-term QA instances) using the mem0 framework to verify that our taxonomy is not overfitted to LoCoMo. As shown in Table 7, applying MemEval to this dataset demonstrates that our 11-class error taxonomy remains orthogonal and comprehensive. The meta-evaluator successfully mapped 252 failures to the established taxonomy, revealing that error distributions vary significantly across benchmarks. In LongMemEval-S, the dominant bottleneck shifts heavily toward the Extraction stage, specifically Missing Critical Information (Label 1.1, 166 cases), particularly within multi-session and temporal reasoning contexts. Furthermore, Utilization-stage failures (Labels 4.1 and 4.2) indicate that even when memories are available, the backbone model may still fail to synthesize them. These findings confirm that the causal tracking logic of MemEval seamlessly adapts to new conversational environments, establishing it as a universal diagnostic tool. A comprehensive stage-wise analysis and detailed breakdown of these error distributions are provided in Appendix D.

5 Conclusion

In this work, we introduce MemEval, a novel chain-of-stage diagnostic framework that fundamentally shifts the evaluation of LLM-based memory systems from monolithic black-box testing to a transparent, causal pipeline. By decomposing the memory lifecycle into Extraction, Update, Retrieval, and Utilization, we established a comprehensive 11-class error taxonomy. To enable scalable and robust diagnosis, we developed a dual-track protocol featuring an automated Ensemble Voting (EV) meta-evaluator, which reliably mimics human root-cause analysis and achieves a strong 81.62% stage-level agreement with expert annotations.

Our granular analysis across diverse memory architectures—including Mem0, A-mem, MemoryOS, and OpenClaw—consistently reveals that Extraction acts as the universal bottleneck. Memory systems frequently suffer from lossy compression at the entry point, irreversibly dropping fine-grained or temporal details that handicap all subsequent operations. Furthermore, extensive scaling experiments demonstrate that merely increasing base model capacity fails to resolve these inherent structural vulnerabilities. Instead of eliminating errors, scaling to stronger backbones primarily shifts the locus of failure from early extraction deficits to downstream reasoning and utilization bottlenecks.

Finally, validation across the LoCoMo and LongMemEval benchmarks confirms that MemEval is architecture-agnostic and adaptable to varying conversational distributions. By piercing through deceptive end-to-end performance metrics to expose localized, stage-specific flaws, MemEval provides the actionable, granular insights required to engineer robust memory systems. We anticipate that this framework will catalyze a paradigm shift in agentic memory research, guiding the field toward targeted, pipeline-aware architectural optimizations rather than brute-force capability scaling.

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.

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 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 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 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, employing a two-stage (segment-to-page) retrieval process in its mid-term memory to balance scope and precision.

Utilization 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 Detailed Consistency Analysis between Human and LLM Annotations

This appendix provides the detailed category-wise breakdown of the alignment between human expert annotations and the diagnostic labels generated by our automated Ensemble Voting (EV) strategy.

As discussed in the main text, the EV framework demonstrates high overall robustness for fault localization. Table 6 details the Exact Match and Stage Match percentages across four distinct question categories from the LoCoMo benchmark. The results highlight that while Temporal and Single-hop tasks achieve high stage-level alignment (87.54% and 83.59%, respectively), complex reasoning tasks such as Multi-hop and Open-domain queries present a stricter challenge for the automated meta-evaluator, reinforcing the necessity of human oversight in these specific scenarios.

C Application of MemEval on other Memory Systems

C.1 A-mem

The diagnostic results for A-mem, visualized in Figures 6 and 7, 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 6), 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 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-

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.

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 6: Consistency between Human and LLM-vote annotations broken down by QA category.

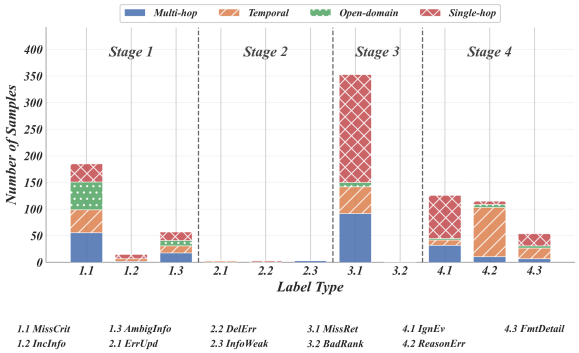

Figure 6: Absolute frequency of LLM-generated diagnostic labels for the A-mem framework across pipeline stages.

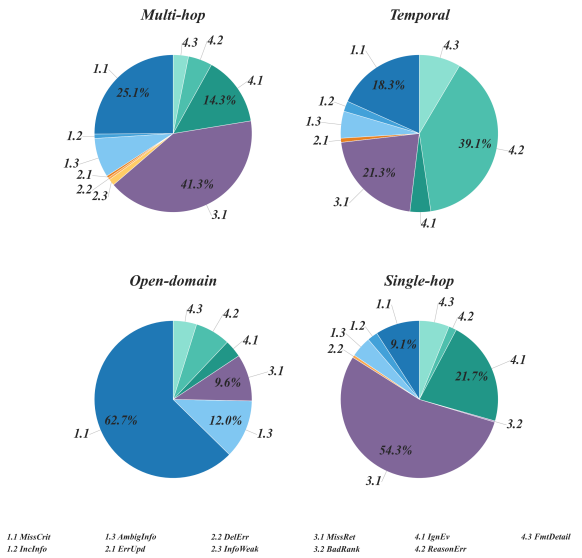

Figure 7: Relative distribution of LLM-generated error labels for the A-mem framework, segmented by LoCoMo QA category.

dependent—suffering from recall issues in factual tasks, encoding loss in open chat, and reasoning gaps in temporal scenarios.

C.2 MemoryOS

The diagnostic landscape for MemoryOS, depicted in Figures 8 and 9, 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.

C.3 OpenClaw

To examine the broader applicability of our observations, we extended our evaluation to include

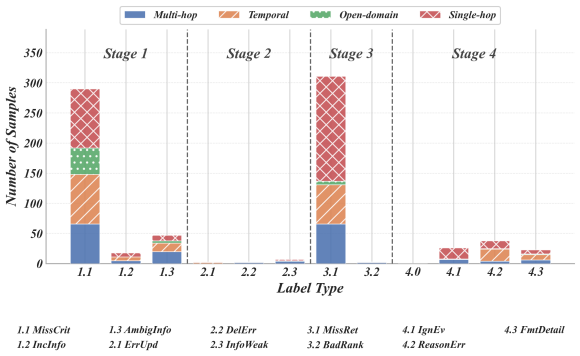

Figure 8: Absolute frequency of LLM-generated diagnostic labels for the MemoryOS framework across pipeline stages.

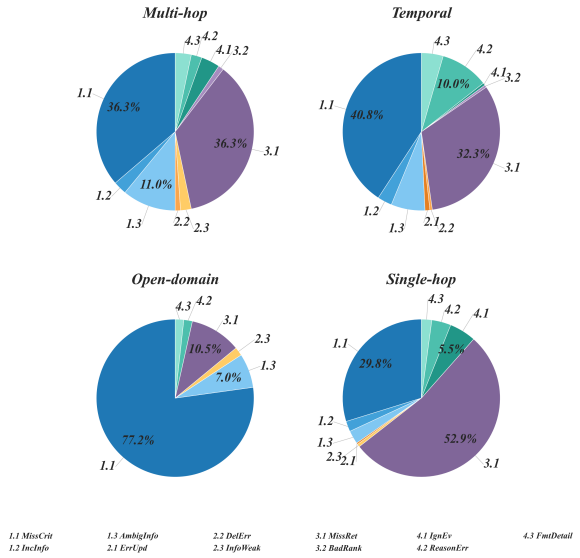

Figure 9: Relative distribution of LLM-generated error labels for the MemoryOS framework, segmented by LoCoMo QA category.

the OpenClaw framework, running MemEval on the LoCoMo benchmark. This experiment encompassed 1,540 QA instances across the four LoCoMo question categories. The results demonstrate that MemEval can be seamlessly applied to OpenClaw without any modifications to the diagnostic taxonomy, confirming the tool's versatility with newer memory systems.

The findings from OpenClaw further underscore the necessity of stage-wise diagnosis. Although OpenClaw achieved a robust overall end-to-end accuracy of 73.70% (1,135 correct samples), MemEval still successfully identified and categorized 405 error cases, revealing clear stage-level failure patterns.

As illustrated by the absolute counts in Figure 10 and the proportional distributions in Figure 11, the

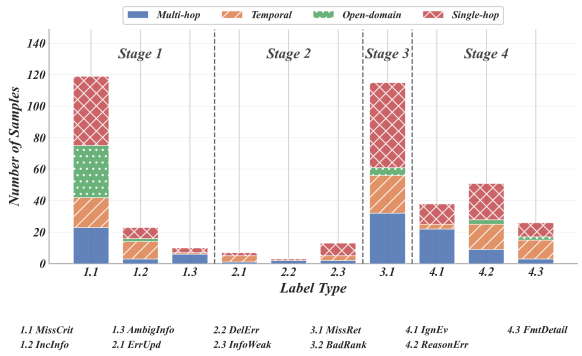

Figure 10: Absolute frequency of LLM-generated diagnostic labels for the OpenClaw framework across pipeline stages.

most frequent error types are Missing Critical Information (Label 1.1, 119 cases) and Missed Retrieval (Label 3.1, 115 cases). This indicates that, even in a recent memory framework, memory extraction (Stage 1) and retrieval (Stage 3) remain the primary bottlenecks. Furthermore, Utilization-stage errors are distinctly visible, including Reasoning Errors (Label 4.2, 51 cases), Ignoring Evidence (Label 4.1, 38 cases), and Format/Detail Errors (Label 4.3, 26 cases).

These findings strengthen our conclusions in two key ways. First, they confirm that MemEval is not limited to the original memory architectures studied in our primary evaluation. Second, they demonstrate that newer frameworks can still exhibit critical, stage-specific failures that are easily obscured by end-to-end accuracy metrics. Consequently, we position MemEval as a robust and necessary diagnostic tool for both established and emerging memory architectures.

D 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 entire LongMemEval dataset. We evaluated the full 500 representative long-term QA instances from the `longmemeval_s` subset.

Table 7 reports the diagnostic results categorized by LongMemEval-S question types. The error columns correspond to the stage-wise error types defined in Section 3.3: Extraction Errors include Missing Critical Information (Label 1.1), Incorrect Information (Label 1.2), and Ambiguous Information (Label 1.3); Update Errors include Erroneous

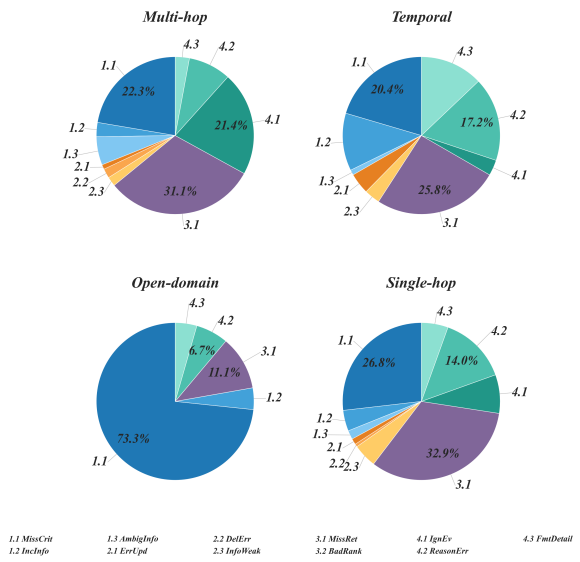

Figure 11: Relative distribution of LLM-generated error labels for the OpenClaw framework, segmented by LoCoMo QA category.

Updates (Label 2.1), Deletion Errors (Label 2.2), and Information Weakening (Label 2.3); Retrieval Errors include Missed Retrieval (Label 3.1) and Unreasonable Ranking (Label 3.2); Utilization Errors include Ignoring Evidence (Label 4.1), Reasoning Errors (Label 4.2), and Format/Detail Errors (Label 4.3).

These results provide stronger evidence that our diagnostic taxonomy and tool can be effectively applied beyond LoCoMo. At the same time, the results also demonstrate that error distributions vary significantly across benchmarks and question categories. We therefore avoid overclaiming generalization and describe our conclusions more cautiously.

In LongMemEval-S, the dominant error is Missing Critical Information (Label 1.1) in the Extraction stage, especially for `multi-session`, `single-session-assistant`, and `temporal-reasoning` questions. This suggests that in cross-session, multi-turn, or temporal-reasoning settings, memory systems often fail to extract the key information needed for subsequent answers. Incorrect Information (Label 1.2) in the Extraction stage is also visible in `knowledge-update` and `temporal-reasoning` questions, indicating that when conversations involve updates or temporal changes, systems may extract not only incomplete but also erroneous memories.

By contrast, Update-stage errors such as Erroneous Updates (Label 2.1), Deletion Errors (Label 2.2), and Information Weakening (Label 2.3) are less frequent overall, although they still reflect local instability in the memory update process. Retrieval-stage Missed Retrieval (Label 3.1) accounts for a smaller portion of the errors in this setting, suggesting that the primary bottleneck is often earlier memory extraction rather than retrieval alone. Finally, Utilization-stage errors such as Ignoring Evidence (Label 4.1) and Reasoning Errors (Label 4.2) further demonstrate that even when relevant memories are correctly extracted and available, the backbone model may still fail to synthesize or utilize them correctly.

E Detailed Case Studies and Root Cause Analysis

To provide a concrete understanding of how cascading errors manifest within the memory pipeline, we present a detailed analysis of four representative cases corresponding to each functional stage (as visualized in Figure 2). For each case, we describe the error manifestation and analyze the underlying root cause.

Case 1: Ext. (1.1 Missing Critical Information) In the first scenario, the dialogue is timestamped September 4, 2022, and the user (James) mentions an event that occurred "Yesterday." During the extraction phase, the system preserves the salient main event (James confessed to Samantha at a theater, and she agreed) but completely omits the relative temporal cue ("yesterday"). Because this anchor is lost, the system cannot deduce that the factual event date is September 3, 2022. When later queried about the date of this event, the model falls back on reasoning from incomplete information, outputting the dialogue's metadata timestamp instead. This error reflects a fundamental tension between memory compression and the uncertainty of future queries. Current memory systems heavily rely on LLM-based, summary-style extraction. LLMs inherently optimize for preserving salient, high-frequency facts while often dropping fine-grained, implicit, or temporal details that appear unimportant at the time of writing. Because the system compresses dialogue content without knowing what specific evidence future questions will require, it inevitably drops details that later become critical.

Case 2: Upd. (2.3 Information Weakening) The original dialogue explicitly lists three specific

Model	QA.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	A	7	8	0	0	0	0	0	0	0	1	0	62	78	16	79.49
	B	48	3	0	1	2	1	4	0	10	2	1	61	133	72	45.86
	C	35	1	0	1	0	1	0	0	2	0	0	16	56	40	28.57
	D	14	0	0	0	0	0	3	0	2	4	0	7	30	23	23.33
	E	6	0	0	0	0	0	1	0	0	1	2	60	70	10	85.71
	F	56	17	0	3	2	1	3	0	3	5	1	42	133	91	31.58
	Total	166	29	0	5	4	3	11	0	17	13	4	248	500	252	49.60

Table 7: Diagnostic results on the LongMemEval-S benchmark using the mem0 framework (powered by GPT-4.1-mini). The **QA.cat** mapping is as follows: A: knowledge-update, B: multi-session, C: single-session-assistant, D: single-session-preference, E: single-session-user, F: temporal-reasoning.

games John played: "Fortnite, Overwatch, and Apex Legends." While the initial memory addition captures this perfectly, a subsequent memory update operation intended to consolidate the user's gaming history weakens the specific entities into a generic phrase: "Fortnite and other games." When the user later asks a specific question about what games John played, the system cannot recall Overwatch and Apex Legends. This failure stems from over-summarization and entity loss during memory consolidation. When merging new dialogue states with existing memories, LLMs tend to abstract information to maintain brevity and reduce token overhead. This causes a lossy compression where specific, low-frequency entities are pruned in favor of general trends, permanently degrading the resolution of the memory store.

Case 3: Ret. (3.1 Missed Retrieval) The memory database contains a precise, factually correct memory about Melanie's family camping trips ("roast marshmallows, tell stories"). However, when a query is issued, the retrieval module ranks topically similar but factually irrelevant memories (e.g., generic statements about hiking and bonding with nature) higher than the target memory. The crucial piece of evidence is squeezed out of the final top-K context window, leading to a hallucinated answer. This highlights the vulnerability of relying solely on coarse-grained semantic similarity (e.g., embedding vectors) for retrieval. High-frequency topical keywords (like "family," "nature," "mountains") can dominate the vector space, causing the system to retrieve broadly related contextual noise while failing to identify the exact factual record required for the specific query.

Case 4: Util. (4.2 Reasoning Errors) In this scenario, the system operates flawlessly in the first three stages: it successfully extracts, maintains, and retrieves the exact correct memory ("Caroline talked about her transgender journey at a school event last week") along with the correct dialogue timestamp ("9 June, 2023"). However, when asked "When did Caroline give a speech at a school?", the generative model blindly copies the dialogue timestamp (9 June 2023) instead of logically deducing that "last week" refers to the week prior. This error exposes the generative limitations of the LLM in bridging temporal or logical gaps. Even provided with perfect context, the model often prioritizes surface-level text matching (copying the explicit metadata date) over multi-step deductive reasoning (anchoring "last week" against the timestamp). It proves that flawless extraction and retrieval are insufficient if the final utilization module lacks the reasoning capacity to correctly synthesize the evidence.

F Human Annotation Cost Analysis

To contextualize the practical need for our automated diagnostic framework, we analyzed the human annotation costs for manual stage-wise error diagnosis. Diagnosing a single instance typically requires 3–5 minutes for a trained expert, translating to 75–125 hours of continuous effort for 1,500 instances. Furthermore, roughly 20% of these samples are ambiguous boundary cases that necessitate an additional 8–12 minutes of multi-expert deliberation per instance to reach a consensus. This makes large-scale, comprehensive manual diagnosis prohibitively expensive and time-consuming. Consequently, MemEval is designed to serve as a scalable preliminary diagnostic tool that efficiently identifies likely error types and sources, thereby reserving valuable human expertise strictly for verification and the review of complex boundary cases.

G 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 8, the pipeline consumes only $\sim$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.

Metric	Value
API Calls	$\sim$1.95 / query
Token Cost	$\sim$2.3k / query
Avg. Latency	11.37s / query

Table 8: Resource consumption metrics for MemEval under the Single-Model Inference (SMI) configuration.

H Detailed Model Performance Breakdown

This appendix provides the fine-grained diagnostic results across all task categories (Multi-hop, Temporal, Open-domain, Single-hop) for both closed-source and open-source models evaluated in this study. Table 9 details the absolute frequency of errors, correct answers, and overall accuracy for each specific stage and task type.

The detailed diagnostic results in Table 9 provide a granular view of how different base models interact with varying task complexities within the Mem0 architecture. Most notably, the data from the open-source Qwen family perfectly illustrates the diminishing returns of parameter scaling in unoptimized memory architectures.

Scaling the base model from 35B (Qwen3.5-35B) to 122B (Qwen3.5-122B) parameters yields an almost identical error fingerprint across all functional stages. The absolute error counts for Multi-hop tasks remain virtually unchanged: early Extraction failures (Label 1.1) drop negligibly from 52 to 47, while downstream Missed Retrievals (Label 3.1) stagnate at 30 versus 31. A similar stagnation is observed in Open-domain tasks, where both models hit exactly 47 extraction errors, limiting their accuracy to a mere 36.46%.

This persistent failure pattern confirms that simply injecting more parametric knowledge or raw reasoning capacity into the base model does not automatically resolve the structural friction within the RAG-based memory pipeline. The core bottleneck remains the systemic inability to reliably extract and link critical information at the entry point of the architecture, an issue that scaling alone cannot fix.

I Prompts for MemEval

I.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.

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.

I.2 Prompt for Extraction Stage

Figure 13 illustrates the prompt designed for the extraction stage. It serves to assess the quality of

Model	QA.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 (%)
Closed-source Models																
GPT-4.1-mini	A	84	1	0	0	0	1	69	2	7	9	3	113	282	169	40.07
	B	72	3	1	10	2	1	31	9	2	13	7	167	321	154	52.02
	C	34	0	0	0	0	0	11	1	2	9	1	39	96	57	40.62
	D	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	A	71	3	4	1	9	1	36	2	22	10	8	167	282	115	40.78
	B	70	3	11	3	12	4	9	3	9	37	20	181	321	140	43.61
	C	39	1	3	0	0	1	5	0	2	6	3	60	96	36	37.50
	D	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
Open-source Models																
Qwen3.5-35B	A	52	3	2	3	0	1	30	0	11	15	6	159	282	123	56.38
	B	88	1	7	1	0	2	4	0	13	0	10	30	321	291	9.35
	C	47	1	0	0	0	0	4	0	2	5	2	35	96	61	36.46
	D	102	12	0	5	0	2	15	0	8	20	17	660	841	181	78.48
	Total	289	17	9	9	0	5	53	0	34	40	35	884	1540	656	57.40
Qwen3.5-122B	A	47	4	3	1	3	6	31	0	7	14	4	162	282	120	57.45
	B	87	1	6	8	2	2	4	0	14	0	20	33	321	288	10.28
	C	47	2	1	0	0	0	1	0	1	5	4	35	96	61	36.46
	D	83	13	2	5	1	3	17	0	11	21	17	668	841	173	79.43
	Total	264	20	12	14	6	11	53	0	33	40	45	898	1540	642	58.31

Table 9: Detailed diagnostic results on the LoCoMo benchmark across closed-source (GPT-4.1-mini, GPT-4o-mini) and open-source (Qwen3.5-35B, Qwen3.5-122B) models. The **QA.Cat** mapping is as follows: A: `Multi-hop`, B: `Temporal`, C: `Open-domain`, D: `Single-hop`. The numerical columns (1.1 to 4.3) map to our stage-level diagnostic taxonomy (e.g., 1.1 for Missing critical information).

the initial memories retrieved by the memory system, specifically identifying whether the extracted content is problematic and classifying the error into a specific category.

I.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.

I.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.

I.5 Prompt for Utilization Stage

Only when Extraction, Update, and Retrieval are verified correct, Figure 16 shows the prompt that assesses whether the LLM fails to utilize these memories and classifies 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.