DIAGNOSING RETRIEVAL VS. UTILIZATION BOTTLENECKS IN LLM AGENT MEMORY

Boqin Yuan
University of California, San Diego
b4yuan@ucsd.edu

Yue Su
Carnegie Mellon University
yuesu@andrew.cmu.edu

Kun Yao
University of North Carolina
kya@unc.edu

ABSTRACT

Memory-augmented LLM agents store and retrieve information from prior interactions, yet the relative importance of *how* memories are written versus *how* they are retrieved remains unclear. We introduce a diagnostic framework that analyzes how performance differences manifest across write strategies, retrieval methods, and memory utilization behavior, and apply it to a 3×3 study crossing three write strategies (raw chunks, Mem0-style fact extraction, MemGPT-style summarization) with three retrieval methods (cosine, BM25, hybrid reranking). On LoCoMo, retrieval method is the dominant factor: average accuracy spans 20 points across retrieval methods (57.1% to 77.2%) but only 3–8 points across write strategies. Raw chunked storage, which requires zero LLM calls, matches or outperforms expensive lossy alternatives, suggesting that current memory pipelines may discard useful context that downstream retrieval mechanisms fail to compensate for. Failure analysis shows that performance breakdowns most often manifest at the retrieval stage rather than at utilization. We argue that, under current retrieval practices, improving retrieval quality yields larger gains than increasing write-time sophistication.

1 INTRODUCTION

A growing body of work equips LLM agents with persistent memory (Packer et al., 2023; Xu et al., 2025; Chhikara et al., 2025). These systems differ mainly in what they store: some keep raw conversation text (Lewis et al., 2020), others extract structured facts with conflict resolution (Chhikara et al., 2025; Xu et al., 2025), and still others compress sessions into summaries (Packer et al., 2023). Recent work has pushed further with inter-memory linking (Xu et al., 2025) and reinforcement learning for end-to-end memory management (Zhou et al., 2025; Wang et al., 2025). Yet a basic question remains open: *does the write strategy actually matter for downstream performance, or do performance differences primarily arise at retrieval?* Current benchmarks (Maharana et al., 2024; Wu et al., 2025) only measure end-to-end accuracy, making it hard to tell whether errors stem from what was stored, how it was retrieved, or how the LLM used the context (Figure 1). We review related work in Appendix A.

We address this with two contributions. First, we propose a **diagnostic probing framework** that sits at the retrieval-to-generation boundary and independently measures retrieval relevance, memory utilization, and failure modes. Second, we conduct a **controlled** 3×3 **factorial study** that crosses three write strategies (raw chunks, Mem0-style extraction, MemGPT-style summarization) with three retrieval methods (cosine similarity, BM25, hybrid reranking); see Appendix B for a detailed discussion of the design rationale. Evaluating on LoCoMo (1,540 non-adversarial questions), we find that retrieval method is the dominant factor: accuracy varies by 20 points across retrieval methods but only 3 to 8 points across write strategies. The cheapest write approach, storing raw chunks with zero LLM calls, matches or outperforms the more expensive alternatives.

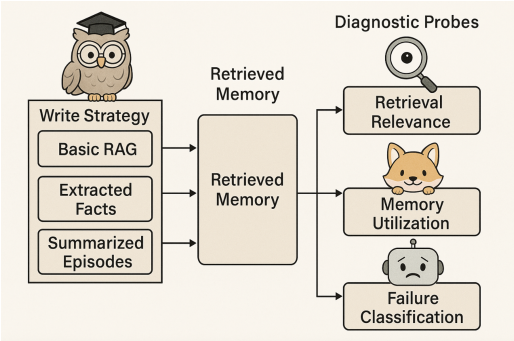

Figure 1: Memory-agent pipeline with three diagnostic probes at the retrieval-to-generation boundary.

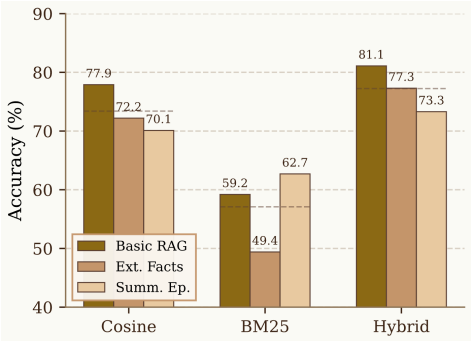

Figure 2: Accuracy across the 3×3 grid (k=5). Retrieval method drives 14–23 point differences; write strategy only 3–8.

2 METHOD

2.1 WRITE STRATEGIES

We evaluate three prompt-based write strategies on LoCoMo (Maharana et al., 2024) (10 multi-session conversations, ∼600 turns each, ∼200 questions per conversation). All configurations use GPT-5-mini as the backbone LLM and `text-embedding-3-small` (1536-d) for embeddings.

Basic RAG stores raw 3-turn conversation chunks with speaker names and timestamps, requiring zero LLM calls at write time. **Extracted Facts** follows Mem0 (Chhikara et al., 2025): an LLM extracts self-contained facts per session, followed by embedding-based matching and LLM conflict resolution (ADD/UPDATE/NOOP). **Summarized Episodes** follows MemGPT (Packer et al., 2023): each session is compressed into a single summary paragraph. Table 3 summarizes computational costs; full prompts and implementation details are provided in Appendix C.

2.2 RETRIEVAL METHODS

We pair each write strategy with three retrieval methods of increasing sophistication, with a default retrieval budget of k=5. **Cosine similarity** returns the top-k memory entries by cosine distance between the query embedding and each stored entry's embedding; this is the default method in most deployed memory systems (Chhikara et al., 2025; Xu et al., 2025) and captures semantic similarity, but may miss lexically relevant entries with different surface forms. **BM25** scores entries by term-frequency-based keyword overlap, complementing embedding-based retrieval by surfacing entries that share exact terms with the query; however, it fails when relevant memories use different vocabulary than the question. **Hybrid+Rerank** first pools the top-$2k$ candidates from both cosine and BM25, then uses GPT-5.2 as an LLM judge to rerank the union down to top-k; this adds one LLM call per query but leverages both semantic and lexical signals, with the reranker resolving disagreements between the two retrieval sources.

2.3 PROBING FRAMEWORK

For each question q with gold answer a^*, we generate an answer a_{mem} *with* retrieved memory (top-k entries in the prompt) and an answer a_{no} *without* any memory, using the same LLM and prompt template. We then apply three diagnostic probes. **Probe 1 (Retrieval Relevance):** An LLM judge evaluates whether each retrieved entry m_i contains information relevant to answering q; we report Retrieval Precision@k = $|\{m_i : \text{relevant}\}|/k$. **Probe 2 (Memory Utilization):** An LLM judge compares a_{mem} and a_{no} against a^* and classifies each question as **Beneficial** (memory improved the answer), **Harmful** (memory worsened it), **Ignored** (answer unchanged), or **Neutral** (answer changed but correctness unaffected). **Probe 3 (Failure Classification):** For incorrect answers, we group failures into three categories: *Retrieval failure* covers cases where the system did not surface

sufficient information to answer the question at inference time. This includes (i) cases where relevant information was not retrieved despite being present in the memory store, and (ii) cases where the stored memories did not contain sufficient detail to support the answer. We group both under retrieval-stage failure because the breakdown manifests at the retrieval-to-generation boundary; *Utilization failure* applies when at least one relevant memory was retrieved but the model still produced an incorrect answer, indicating a downstream reasoning problem; and *Hallucination* captures the rare cases where the model's answer directly contradicts the content of its own retrieved memories.

3 RESULTS

3.1 WRITE STRATEGY HAS MINIMAL IMPACT

Table 1 shows that write strategy accounts for only 3–8 accuracy points within any retrieval column. Basic RAG, which stores raw chunks with zero LLM calls, achieves 77.9% under cosine and 81.1% under hybrid, matching or exceeding both Extracted Facts and Summarized Episodes across all retrieval methods. The one exception is BM25, where Summarized Episodes (62.7%) edges out Basic RAG (59.2%), likely because keyword retrieval benefits from the denser vocabulary of compressed summaries. Overall, the more expensive write strategies do not justify their cost: lossy compression discards conversational details that the backbone LLM can otherwise leverage directly (Figure 2).

Table 1: Token F1 and LLM-as-Judge accuracy across all nine configurations (k=5, 1,540 non-adversarial questions). Accuracy spread across retrieval methods (14–23 pts) consistently exceeds spread across write strategies (3–8 pts). Bold = best write strategy per retrieval method.

	Token F1			**Accuracy (%)**		
	Cos	**BM25**	**Hyb**	**Cos**	**BM25**	**Hyb**
Basic RAG	**0.232**	**0.184**	**0.240**	**77.9**	59.2	**81.1**
Extracted Facts	0.211	0.146	0.220	72.2	49.4	77.3
Summ. Episodes	0.197	0.173	0.202	70.1	**62.7**	73.3
Avg	0.213	0.168	0.221	73.4	57.1	77.2

3.2 RETRIEVAL METHOD DOMINATES

Switching retrieval method shifts accuracy by 14–23 points, dwarfing the write strategy effect. Hybrid reranking averages 77.2% across write strategies, compared to 73.4% for cosine and 57.1% for BM25. This 20-point gap between hybrid and BM25 holds regardless of what is stored, confirming that surfacing the right context matters far more than how that context was formatted. Retrieval precision is near-perfectly correlated with downstream accuracy (r=0.98, Figure 3), providing strong evidence that retrieval quality most strongly correlates with downstream performance. We also note that token F1 is largely insensitive to these accuracy differences (0.221 vs. 0.168 for a 20-point accuracy gap), underscoring the importance of LLM-judge evaluation over surface-level overlap metrics.

3.3 FAILURE ANALYSIS CONFIRMS RETRIEVAL AS THE BOTTLENECK

Table 2 and Figure 4 break down where failures actually occur. Retrieval failure is the dominant error mode, accounting for 11–46% of all questions depending on configuration. Under BM25 with Extracted Facts, retrieval failure alone reaches 46.3%, nearly matching the total error rate. Hybrid reranking cuts retrieval failures roughly in half (e.g., 35.3% to 11.4% for Basic RAG). By contrast, utilization failures remain stable at 4–8% and hallucinations at 0.4–1.4% regardless of configuration, indicating that the LLM reliably uses relevant context when it is provided. The utilization probe reinforces this: beneficial rates reach 79.0% under Basic RAG + Hybrid, meaning the model improves its answer four out of five times when given good retrievals. These patterns consistently point to the same conclusion: performance breakdowns most often manifest at the retrieval stage rather than at utilization.

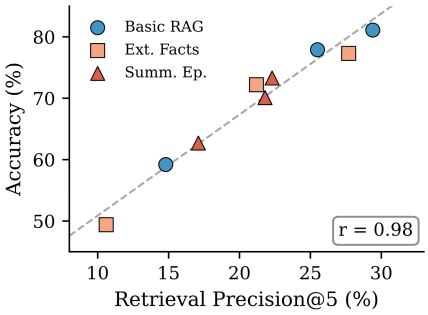

Figure 3: Precision@5 vs. accuracy (r=0.98). Each point is one of nine configurations.

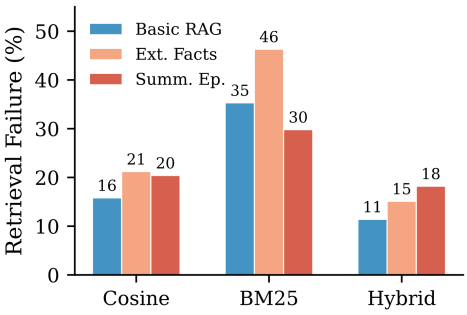

Figure 4: Retrieval failure rate by retrieval method. Hybrid reranking cuts retrieval failures by half or more across all write strategies.

Table 2: Diagnostic probes across all nine configurations (%). Probe 1: retrieval precision. Probe 2: memory utilization. Probe 3: failure modes as percentage of all questions (remainder = correct). See Section 3.3.

		Probe 1	**Probe 2: Utilization**		**Probe 3: Failure Mode**		
		Precision	**Beneficial**	**Ignored**	**Retrieval**	**Utilization**	**Halluc.**
Basic RAG	Cosine	25.5	75.7	9.2	15.8	5.4	1.0
	BM25	14.8	56.5	**24.8**	35.3	5.1	0.4
	Hybrid	**29.4**	**79.0**	**6.6**	**11.4**	6.2	1.2
Ext. Facts	Cosine	21.2	70.4	9.7	21.2	5.9	0.6
	BM25	10.6	46.2	**29.2**	**46.3**	3.9	0.5
	Hybrid	**27.7**	**75.3**	7.2	15.1	6.9	0.7
Summ. Ep.	Cosine	21.8	67.9	11.4	20.4	8.1	1.4
	BM25	17.1	60.4	18.1	**29.8**	6.4	1.0
	Hybrid	**22.3**	**70.1**	9.9	18.2	7.1	1.4

4 Discussion and Conclusion

Our results show that retrieval quality has a substantially larger impact on performance than write strategy. Fact extraction and summarization inevitably discard contextual nuances useful at inference time, while raw chunking preserves the full signal with zero LLM calls and matches or outperforms costlier alternatives. Hybrid reranking confirms that most failures stem from irrelevant retrieval rather than memory construction deficiencies; when relevant context is surfaced, the model uses it effectively. More broadly, these findings suggest that modern LLMs already possess strong contextual reasoning, and our results suggest that performance differences are more strongly associated with information selection than with representation choices. This reframes design priorities for memory-augmented agents: progress may depend more on retrieval precision, reranking, and query understanding than on increasingly complex write pipelines.

Limitations. Our study evaluates a single backbone model (GPT-5-mini) on one benchmark (LoCoMo) with a fixed retrieval budget (k=5), limiting generality. The write strategies are prompt-based reimplementations rather than fully learned memory systems, and results may differ for reinforcement learning approaches that jointly optimize writing and retrieval. The advantage of raw chunking may also diminish under tighter context budgets where compression is required to fit within model input limits. Additionally, answer correctness and failure classification rely on LLM judges. Extending this analysis across models, datasets, retrieval budgets, and learned memory systems would strengthen the robustness of our conclusions.

ACKNOWLEDGMENTS

We thank our annotators for their work on failure classification and answer validation. Experiments were supported by API credits for OpenAI models inference. Portions of the manuscript were edited with assistance from a large language model to improve clarity and presentation. All experimental design, analysis, and conclusions were developed by the authors.

REFERENCES

Prateek Chhikara, Deshraj Khare, and Deshraj Tariq. Mem0: Building production-ready memory layer for LLM applications. *arXiv preprint arXiv:2504.19413*, 2025.

Yuanzhe Hu, Yu Wang, and Julian McAuley. Evaluating memory in llm agents via incremental multi-turn interactions, 2025. URL https://arxiv.org/abs/2507.05257.

Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. Realtalk: A 21-day real-world dataset for long-term conversation, 2025. URL https://arxiv.org/abs/2502.13270.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

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

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-
$alpha$: Learning memory construction via reinforcement learning, 2025. URL https://arxiv.org/abs/2509.25911.

Di Wu, Hongwei Jiang, Wenhao Wu, et al. LongMemEval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2025.

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

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory, 2023. URL https://arxiv.org/abs/2305.10250.

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

A Related Work

Memory systems for LLM agents span a design spectrum from raw storage to learned management. MemGPT (Packer et al., 2023) uses recursive summarization; Mem0 (Chhikara et al., 2025) extracts facts with conflict resolution; A-MEM (Xu et al., 2025) adds inter-memory linking; and MemoryBank (Zhong et al., 2023) combines raw logs with hierarchical summaries and forgetting curves. MEM1 (Zhou et al., 2025) and Mem-α (Wang et al., 2025) learn memory management via reinforcement learning, falling outside our prompt-based scope. Several benchmarks evaluate long-term memory, including LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2025), MemoryAgentBench (Hu et al., 2025), and RealTalk (Lee et al., 2025), but all measure downstream accuracy without decomposing *why* a system fails. Our probing framework fills this gap.

B Experimental Design and Rationale

Dataset Details. LoCoMo contains 10 multi-session conversations (approximately 600 turns each) with roughly 200 questions per conversation spanning single-hop, multi-hop, temporal reasoning, and open-domain categories (adversarial excluded). We evaluate on 1,540 non-adversarial questions.

Memory-augmented agents are often evaluated as end-to-end systems, which obscures where errors arise. A single accuracy number conflates three factors: whether the write strategy preserved the relevant information, whether retrieval surfaced it, and whether the model used it correctly. While our probes separate retrieval-stage breakdowns from utilization errors, we do not independently measure write-time information loss; instead, we analyze how failures manifest at inference time. Our experimental design therefore enables a controlled comparison of write strategies and retrieval methods, allowing us to identify which stage most strongly correlates with downstream performance.

We adopt a 3×3 factorial design that crosses three write strategies with three retrieval methods (Table 3). The write strategies span the information preservation spectrum: Basic RAG stores raw conversation chunks with no processing; Extracted Facts compresses sessions into structured factual statements with conflict resolution; and Summarized Episodes condenses each session into a narrative summary. The retrieval methods span common approaches: cosine similarity for semantic matching, BM25 for lexical matching, and Hybrid+Rerank, which combines both signals with an LLM reranking step. Evaluating all nine combinations allows us to measure the independent effect of writing and retrieval, as well as potential interactions between them.

All configurations use the same backbone model, embedding model, retrieval budget, and benchmark questions to control for implementation variance. We then apply our three diagnostic probes to every setting, enabling analysis of not only overall performance differences but also the specific failure modes driving them.

Table 3: Experimental design. We cross three write strategies (what is stored) with three retrieval methods (how it is retrieved), yielding a 3×3 evaluation grid.

	Component	**Mimics / Mechanism**	**LLM Calls**
Write	Basic RAG	Store raw 3-turn chunks	None
	Extracted Facts	Mem0-style fact extraction + conflict resolution	1+/session
	Summ. Episodes	MemGPT-style session summarization	1/session
Retrieval	Cosine Similarity	Embedding similarity	None
	BM25	Keyword matching	None
	Hybrid+Rerank	Cosine $\cup$ BM25, LLM rerank to top-k	1/query

C MEMORY PROMPTS

C.1 FACT EXTRACTION (EXTRACTED FACTS STRATEGY)

Extraction Prompt

```
You are a memory extraction agent.  Given a conversation session
between two people, extract ALL important facts, events,
preferences, and details mentioned.
For each fact, provide:
- fact:  A concise, self-contained statement (should make sense
without the original context) - speakers:  Which speaker(s) this
fact is about - type:  One of [event, preference, relationship,
plan, personal_detail, opinion]
Conversation:  {conversation}
Return a JSON object with key "facts" containing a list of
extracted facts.  Example format:  "facts":  [ "fact":  "Alice
adopted a golden retriever named Max in March", "speakers":
["Alice"], "type":  "event", ...]
Extract every meaningful piece of information.  Be thorough,
missing a fact means it's lost forever.
```

C.2 CONFLICT RESOLUTION (EXTRACTED FACTS STRATEGY)

Conflict Resolution Prompt

```
You are a memory management system.  A new fact has been extracted
from a conversation.  You must decide how to handle it relative to
existing memories.
New fact:  {new_fact}
Existing similar memories:  {existing_memories}
Decide one of:  - ADD: The new fact contains genuinely new
information not covered by existing memories.  - UPDATE: The new
fact updates/supersedes one of the existing memories (provide the
ID to update).  - NOOP: The new fact is redundant --- the
information is already fully captured.
Return JSON:

{"action": "ADD" | "UPDATE" | "NOOP",
 "target_id": "<id of memory to update, only if UPDATE>",
 "reason": "brief explanation"}
```

C.3 SESSION SUMMARIZATION (SUMMARIZED EPISODES STRATEGY)

Summarization Prompt

```
You are a memory summarization agent.  Summarize the following
conversation session into a detailed but concise summary.
Your summary MUST capture:  1.  All key events, updates, or changes
in either speaker's life 2.  Any plans, commitments, or intentions
mentioned 3.  Preferences, opinions, or emotional states expressed
4.  Temporal markers (dates, "last week", "tomorrow", etc.)  5.
Any references to previous conversations or shared history
Write the summary as a coherent paragraph.  Include speaker names.
Do NOT omit details, every piece of information matters for future
recall.
Conversation session ({timestamp}):  {conversation}
```

```
Summary:
```

C.4 QUESTION ANSWERING PROMPTS

QA With Memory Prompt

```
You are answering questions about a long-running conversation
between two people.  You have access to retrieved memory entries
from past conversation sessions.
Retrieved memories:
{memories}
Question:  {question}
Based on the retrieved memories, provide a concise and accurate
answer.  If the memories don't contain enough information, say so,
but still try your best based on what's available.  Keep the answer
brief and factual.
```

QA Without Memory Prompt (Control Condition)

```
You are answering questions about a conversation between two
people.  You do NOT have access to any conversation history or
memory.
Question:  {question}
Try your best to answer based on general knowledge.  If you cannot
answer without specific conversation context, say "I don't have
enough information to answer this question." Keep the answer brief.
```

C.5 PROBE 1: RETRIEVAL RELEVANCE

Relevance Judge Prompt

```
You are evaluating whether a retrieved memory entry is relevant to
answering a question.
Question:  {question}
Gold answer:  {gold_answer}
Memory entry:  {memory_content}
Is this memory entry relevant to answering the question?  A memory
is relevant if it contains information that could directly help
produce the correct answer.
Return JSON: {"relevant":  true/false, "reason":  "brief
explanation"}
```

C.6 PROBE 2: MEMORY UTILIZATION

Utilization Judge Prompt

```
You are comparing two answers to the same question.
Question:  {question}
Gold (correct) answer:  {gold_answer}
Answer A (with memory):  {answer_with}
Answer B (without memory):  {answer_without}
Evaluate:  1.  Are the two answers semantically the same?  (i.e.,
they convey the same information, ignoring minor wording
differences) 2.  Is Answer A (with memory) correct or closer to the
```

```
gold answer? 3. Is Answer B (without memory) correct or closer to
the gold answer?
Return JSON: {
  "same_answer": true/false,
  "answer_with_correct": true/false,
  "answer_without_correct": true/false,
  "explanation": "brief explanation"
}
```

C.7 PROBE 3: FAILURE CLASSIFICATION

Failure Classification Prompt

```
You are analyzing a memory-augmented QA system's failure.
Question: {question}
Gold (correct) answer: {gold_answer}
System's answer: {system_answer}
Was the system's answer correct? {is_correct}
Retrieved memories (these were provided to the system):
{retrieved_memories}
Relevance of each memory:
{relevance_judgments}
Classify this case into ONE of these categories: -
"retrieval_failure": The system failed to retrieve relevant
memories, either because they don't exist in the store or retrieval
ranked them poorly. - "utilization_failure": At least one relevant
memory was retrieved, but the system failed to use it correctly in
generating the answer. - "hallucination": The system's answer
directly CONTRADICTS information in the retrieved memories. -
"correct": The system answered correctly.
Return JSON: {
  "failure_category": "one of the categories above",
  "explanation": "1-2 sentence explanation of what went wrong",
  "key_evidence": "the specific memory content that was relevant
but ignored, or the contradiction, if applicable"
}
```

D HUMAN ANNOTATION AND LLM JUDGE VALIDATION

We validate our LLM-as-judge approach for both answer correctness (Probe 2) and failure classification (Probe 3). For answer correctness, we sampled 200 questions stratified by LLM judgment and had human annotators independently label each answer as correct or incorrect. For failure classification, we sampled 200 incorrect answers and had annotators assign one of three failure categories: *Retrieval failure*, *Utilization failure*, or *Hallucination*. Annotators see the question, gold answer, system answer, and all retrieved memories for each case.

D.1 ANSWER CORRECTNESS VALIDATION

Table 4 shows the confusion matrix between LLM judge and human annotations for answer correctness. The LLM judge achieves 92% accuracy with substantial agreement (Cohen's $\kappa = 0.82$), validating its use for measuring answer accuracy in Table 1.

Table 4: Confusion matrix: LLM judge vs. human annotation for answer correctness (200 sampled questions, stratified by LLM judgment). Cohen's $\kappa = 0.82$ indicates substantial agreement.

	Human: Correct	**Human: Incorrect**	**Total**
LLM: Correct	129	9	138
LLM: Incorrect	7	55	62
Total	136	64	200

D.2 Failure Mode Classification Validation

Table 5 shows the confusion matrix for failure mode classification on incorrect answers. The LLM judge achieves 85% accuracy, with most confusion between *retrieval failure* and *utilization failure*. This reflects the inherent difficulty in distinguishing whether the system failed to retrieve the right information or failed to use correctly-retrieved information—a boundary that is often ambiguous even for human judges.

Table 5: Confusion matrix: LLM judge vs. human annotation for failure mode classification

Human \ LLM	**Retrieval**	**Utilization**	**Hallucination**	**Total**
Retrieval Failure	147	7	0	154
Utilization Failure	16	23	0	39
Hallucination	0	1	6	7
Total	163	31	6	200