FROM LOSSY TO VERIFIED: A PROVENANCE-AWARE TIERED MEMORY FOR AGENTS

Qiming Zhu* **Shunian Chen** **Rui Yu** **Zhehao Wu** **Benyou Wang**
School of Data Science, The Chinese University of Hong Kong, Shenzhen
qimingzhu@link.cuhk.edu.cn

ABSTRACT

Long-horizon agents often rely on write-time summaries to keep interaction histories manageable. But compression happens before the system knows what a future query will depend on. As a result, summary-only memory can remain topically relevant while omitting the query-critical detail required for a faithful answer. Always grounding on raw logs avoids this failure mode, but treating raw history as the default evidence source is costly and often unnecessary.

We argue that long-horizon agent memory should therefore support *inference-time evidence allocation*: for each query, the system should use the cheapest evidence that is still sufficient for faithful and traceable answering. We instantiate this principle in **TierMem**, a provenance-linked two-tier memory framework with three components: summary-first retrieval, selective escalation to immutable raw logs, and verified write-back of evidence-backed findings. Across two long-horizon memory benchmarks, TierMem improves the accuracy–efficiency frontier over summary-only memory while substantially reducing the cost of always-raw grounding. On LoCoMo, TierMem reaches 0.851 accuracy versus 0.873 for a raw-grounded, while reducing average input tokens by 54.1% and latency by 60.7%. Code is available at https://github.com/FreedomIntelligence/Tiermem.

1 INTRODUCTION

Long-horizon language agents increasingly need to answer questions grounded in interaction histories that span weeks or months Chen et al. (2023); Liu et al. (2023). In this setting, a memory system is only useful if it satisfies three requirements at once: it must preserve answer-critical evidence, keep that evidence traceable to source interactions, and do so at acceptable latency and cost. Existing agent-memory systems usually obtain efficiency through write-time compression, storing summaries, facts, or other compact memory units instead of the full interaction history Chhikara et al. (2025); Fang et al. (2025). This design works when coarse memory is enough. It becomes brittle when a later query depends on the exact detail that compression did not preserve.

The core difficulty is a *write-before-query asymmetry*—a query-agnostic compression mismatch. Memory is written before the system knows what a future query will require, yet the eventual answer is judged against query-specific evidence. A summary can therefore remain topically relevant while still dropping the decisive detail: an allergy becomes a generic preference, a named entity becomes a role description, or a time constraint disappears altogether. When this happens, the system may still produce a plausible answer, but it can no longer justify that answer from preserved evidence.

A straightforward response is to bypass compression and always ground answers in raw interaction logs Asai et al. (2023); Yan et al. (2025); Du et al. (2025). That restores an authoritative source of truth, but it is too expensive to be a default policy. Many queries are answerable from high-level memory alone, so reading raw logs every time wastes tokens and latency. The practical question is therefore not whether summaries or raw logs are universally better. The question is how a memory system should allocate evidence *at inference time*: answer cheaply when summaries are sufficient, and selectively verify against raw evidence when they are not.

*Corresponding author: qimingzhu@link.cuhk.edu.cn

We argue that this is the right abstraction for long-horizon agent memory. Rather than treating memory as fixed compression alone, a memory system should support *selective verification under unknown future queries*. We instantiate this principle in **TierMem**, a provenance-linked two-tier memory framework. TierMem stores compact summary units in a fast Tier-1, keeps immutable raw pages in Tier-2 as the authoritative record, retrieves summaries first by default, selectively escalates to raw evidence when summaries are insufficient, and writes evidence-backed findings back to Tier-1 with provenance links so future queries can often stay on the cheap path.

Across **LoCoMo** and **LongMemEval**, this design improves the accuracy–efficiency frontier. On **LoCoMo**, TierMem achieves **0.851** accuracy, close to the **0.873** raw-grounded, while reducing average input tokens by **54.1%** and latency by **60.7%**. On **LongMemEval**, where multi-session reasoning, temporal reasoning, knowledge updates, and abstention place greater pressure on fine-grained evidence Wu et al. (2025), TierMem narrows the gap between summary-only memory and always-raw grounding by selectively escalating evidence-insufficient queries.

Contributions.

1. We identify **write-before-query asymmetry** as a structural problem in long-horizon agent memory and formulate memory access as an **inference-time evidence allocation** problem.
2. We introduce **TierMem**, a provenance-linked two-tier memory framework with summary-first retrieval, selective escalation, and verified write-back.
3. We show on two long-horizon memory benchmarks that TierMem improves the accuracy–efficiency trade-off, and we isolate the contributions of provenance linking, selective verification, and write-back through targeted analyses.

2 Problem Diagnosis

The central challenge in long-horizon agent memory is not memory size alone. It is the mismatch between *query-agnostic compression* and *query-specific evidence requirements*. Benchmarks such as LoCoMo and LongMemEval make this mismatch visible because they evaluate multi-session conversational memory. This section diagnoses the failure mode and derives the design requirements that motivate TierMem.

2.1 The problem is not memory length, but evidence mismatch

Write-time compression is useful because it creates a cheap memory substrate for later retrieval. But it is necessarily performed before future queries are known. This means the writer must decide what to keep without knowing which detail will later become answer-critical. The resulting summary can be semantically related to a future query while still being unusable as evidence for that query.

This distinction between *topic relevance* and *evidence sufficiency* is the key diagnostic. A summary may mention the right person, event, or discussion while omitting the exact value, date, entity, negation, or update that the eventual answer depends on. In long-horizon interaction histories, later questions often target precisely these details: exceptions, revisions, temporal qualifiers, or named entities that were minor at write time but decisive at answer time.

2.2 Summary-only failures are often structural, not random

The failure pattern is visible in our LoCoMo analysis. On LoCoMo, the summary-only TierMem variant makes **378** errors out of **1540** queries (**24.5%**). Among these errors, **41.3%** are due to missing information and **28.8%** are due to over-generalization, meaning that **70.1%** of failures arise because the right evidence is absent or too abstract in the summary tier. These are not arbitrary model mistakes; they are the characteristic failure modes of compression performed before future query requirements are known.

LongMemEval shows the same pattern at benchmark scale. Summary-centric systems incur high unverifiable omission rates, ranging from **16.8%** to **29.6%** in our experiments. This is consistent with the benchmark design: LongMemEval explicitly stresses information extraction, multi-session

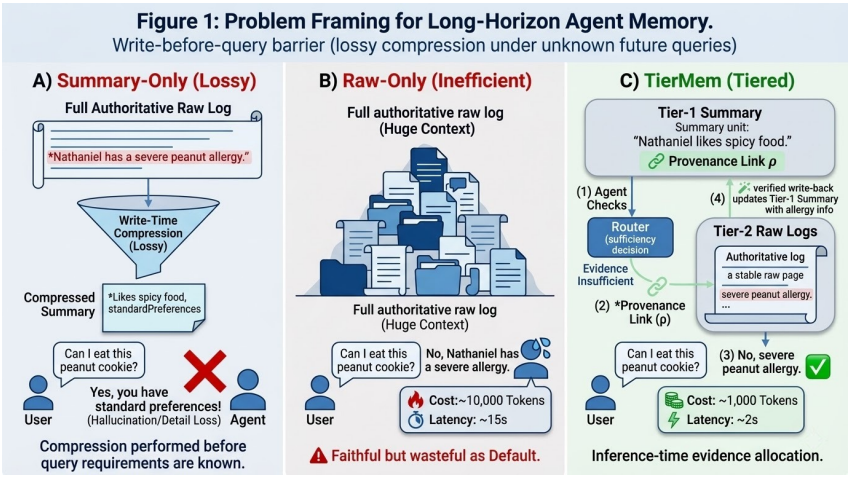

Figure 1: Problem framing for long-horizon memory. Summary-only memory is cheap but lossy, always-raw grounding is faithful but expensive, and TierMem combines summary-first retrieval with selective verification and evidence-backed write-back.

reasoning, temporal reasoning, knowledge updates, and abstention, all of which increase the chance that answer-critical evidence is finer-grained than the compressed memory representation.

What matters next is whether these failures are *recoverable*. If a substantial fraction of summary-only errors can be repaired once the system consults raw evidence, then the right response is not to abandon summaries altogether. The right response is to support selective verification.

2.3 Always-raw grounding is faithful, but wasteful as a default

At the other extreme, one can answer every query from raw interaction logs. This avoids compression loss and provides a stronger grounding path, but it is an inefficient default. On LoCoMo, our always-raw reference uses **7398.4** average input tokens per query, compared with **260.1** for summary-only access. On LongMemEval, the same pattern holds: **7298** versus **380** average input tokens per query. Latency follows the same trend.

These costs matter even when a model can technically accept long contexts. Long inputs are not free simply because they fit in the context window: effective use of evidence can still degrade when relevant details are buried, diffuse, or poorly positioned Liu et al. (2023). For many queries, raw grounding is therefore unnecessary overhead rather than necessary evidence.

2.4 Design requirements

The diagnosis above suggests three requirements for long-horizon agent memory.

(1) Cheap default path. The system should answer from compact memory whenever that evidence is sufficient.

(2) Selective verification against an authoritative source. The system should preserve access to immutable raw evidence and escalate only when summaries are insufficient.

(3) Amortization through evidence-backed write-back. When a costly escalation uncovers decisive evidence, the system should consolidate that finding back into cheap memory while preserving provenance so future queries need not pay the same raw-access cost again.

These requirements point to a multi-tier memory architecture. The problem is not choosing summaries or raw logs once and for all. The problem is supporting *inference-time evidence allocation* under the write-before-query asymmetry.

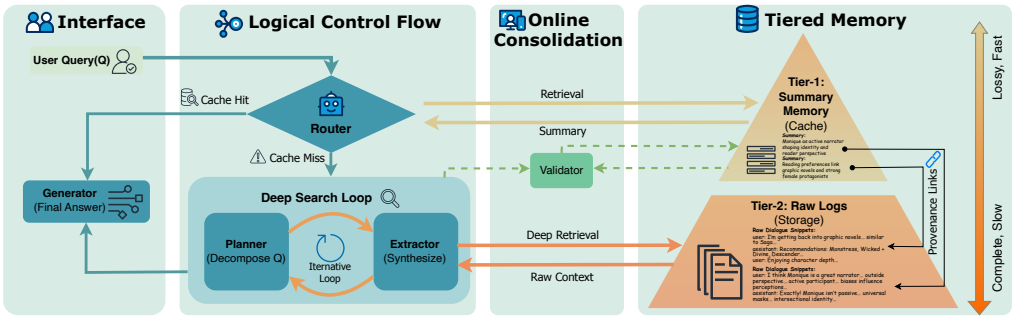

Figure 2: **TierMem overview.** TierMem maintains a provenance-linked two-tier memory hierarchy. Tier-1 is a fast summary index whose entries store compact summaries and provenance links ρ to supporting Tier-2 raw pages in an immutable paged log. Given a query, TierMem retrieves Tier-1 evidence and applies a lightweight sufficiency router to choose between a *fast path* (answer from summaries) and an *escalated path* (consult linked raw pages, then perform bounded additional retrieval if needed). After escalation, TierMem can optionally write evidence-backed findings back to Tier-1 while preserving provenance links.

3 TierMem

TierMem instantiates the design principle above. It has three core components: **summary-first retrieval**, **selective escalation to raw evidence**, and **verified write-back**. Router training matters only insofar as it implements the sufficiency decision that connects these components; it is not the main contribution of the system.

3.1 Architecture

Tier-1 summary tier ($\mathcal{M}_S$). Tier-1 stores compact memory units for fast retrieval. Each entry contains a concise summary, a dense embedding for retrieval, and **provenance links** ρ that point to the supporting Tier-2 raw pages. Tier-1 is the default retrieval substrate because many queries can be answered from compact memory alone.

Tier-2 raw tier ($\mathcal{M}_R$). Tier-2 stores the full interaction history as immutable, fixed-size pages. These pages serve as the authoritative record. Immutability matters because it keeps evidence references stable even as the memory grows, allowing summary entries and write-back updates to remain traceable to source pages.

Provenance-linked construction. Whenever a raw page is sealed in Tier-2, TierMem synthesizes the corresponding Tier-1 summary entries and records provenance links to the source page identifiers. This design ensures that every compact memory unit is born with an explicit path back to raw evidence.

3.2 Inference protocol

Given a query, TierMem chooses the minimum evidence granularity that is sufficient for a faithful answer.

Step 1: Summary-first retrieval. TierMem retrieves the top-k Tier-1 summary units relevant to the query. These summaries are cheap to access and often sufficient for high-level questions.

Step 2: Sufficiency decision. A lightweight **sufficiency router** examines the query together with the retrieved summaries and decides whether the current evidence is enough. If the answer can be supported directly from summaries, TierMem stays on the fast path. If not, TierMem enters the

escalated path. The role of this router is narrow but important: it decides whether summary evidence is sufficient, not how to solve long-horizon memory in general.

Step 3: Provenance-guided escalation. When summaries are insufficient, TierMem first consults the Tier-2 raw pages already linked to the retrieved summaries. This yields a high-recall warm start because those pages are where the compact evidence came from. If that is still insufficient, TierMem performs a bounded additional retrieval procedure over Tier-2. In the main experiments, this loop is capped at $T_{\max} = 3$ steps.

3.3 VERIFIED WRITE-BACK

After a successful escalation, TierMem can consolidate the newly recovered detail into Tier-1. We call this **verified write-back**. Here, *verified* means **evidence-backed with explicit raw-page provenance**, not externally fact-checked against a separate ground-truth source. The write-back controller either adds a new Tier-1 unit, updates an existing one, or skips the write if the evidence is already covered.

This operation is critical to the systems story. Selective verification would be less useful if the system had to pay the same raw-access cost every time. Write-back turns one expensive escalation into future cheap access, while provenance links preserve the path back to supporting evidence.

3.4 ROUTER TRAINING OVERVIEW

We construct router training data from the *Long-Range Understanding (LRU)* competency in MemoryAgentBench Hu et al. (2025). For each query, we compare a summary-only answer with a raw-groundedanswer. The router is trained to predict ANSWER when summary evidence is sufficient and ESCALATE when raw grounding recovers detail that summaries failed to preserve. We initialize the router with supervised distillation from a stronger teacher and then apply cost-aware alignment so the controller learns to escalate only when the expected gain in fidelity justifies the extra retrieval cost.

The full SFT labels, teacher prompts, GRPO reward definition, and hyperparameters are moved to Appendix D. In the main paper, the important fact is simply that the router is a *lightweight sufficiency controller* serving the larger memory architecture.

4 EXPERIMENTAL SETUP

We evaluate TierMem along four dimensions: overall accuracy–efficiency trade-off, the structural value of provenance links, amortization from write-back, and the behavior of the sufficiency controller.

4.1 BENCHMARKS

LoCoMo. LoCoMo evaluates conversational memory over long interaction histories with question types including single-hop recall, open-domain answers, temporal queries, and multi-session reasoning Maharana et al. (2024).

LongMemEval. LongMemEval evaluates long-term memory with information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention Wu et al. (2025). This makes it a good fit for our main claim: long-horizon memory systems must support both efficient coarse recall and query-specific recovery of fine-grained evidence.

4.2 SYSTEMS COMPARED

We compare against representative summary-centric systems (**Mem0**, **LightMem**, **O-mem**, **MemOS**) and raw/controller-centric systems (**MemR3**, **GAM**) Chhikara et al. (2025); Fang et al. (2025); Wang et al. (2025); Li et al. (2025); Du et al. (2025); Yan et al. (2025).

For TierMem, we report four variants: **TierMem (summary-only)**, **TierMem (raw-grounded)**, **TierMem (router)**, and **TierMem (router, GPT-4.1-mini)** as a stronger routing reference. To

Table 1: **Main results on LoCoMo.** The main paper keeps only the metrics needed to show the systems trade-off. Subtype breakdowns, F1, output tokens, and router token decomposition are in appendix A.1.

System	Acc. ↑	UOR ↓	Tok_{in} ↓	Lat. ↓
Mem0	0.684	23.3	1085.9	20.32
LightMem	0.773	14.7	1159.5	27.69
O-mem	0.764	16.3	2178.1	1.16
MemOS	0.753	17.1	2692.3	10.80
MemR3	0.860	–	4585.7	12.30
GAM	0.869	–	15773.7	21.85
TierMem (summary-only)	0.755	15.4	**260.1**	**1.24**
TierMem (raw-grounded)	**0.873**	–	7398.4	17.18
TierMem (router)	0.851	–	3980.4	6.76
TierMem (router, 4.1-mini)	0.851	–	3620.9	6.40

isolate the memory-access policy from model choice, all TierMem variants share the same generator, embedding model, reranker, and retrieval stack. The only differences are whether Tier-2 is consulted and how escalation is triggered.

Following the benchmark protocol, the main benchmark evaluation **disables online write-back** for all methods. Write-back is evaluated separately in a replay study.

4.3 Metrics

Answer quality. Our primary metric is benchmark **LLM-judge accuracy**. We also report **Unverifiable Omission Rate (UOR)**, the fraction of queries where a summary-based method fails because answerable evidence is missing from summaries but available to the raw-grounded.

Efficiency. We report **average input tokens per query** and **average latency per query**. Output-token breakdowns, subtype results, and router-specific token decomposition are moved to the appendix.

4.4 Implementation and fairness notes

All TierMem variants use GPT-4.1-mini as the answer generator in the main comparison. Escalation uses a bounded raw retrieval loop with $T_{\max} = 3$ unless otherwise stated. Latency is reported in the QA phase only (retrieval + generation + routing when applicable).

5 Main Results

The main result is that TierMem improves the accuracy–efficiency frontier relative to static summary-only memory while avoiding the cost of always-raw grounding. In other words, the system behaves as intended: cheap by default, stronger evidence when needed.

5.1 LoCoMo

Table 1 shows the main LoCoMo comparison. TierMem (summary-only) is the cheapest variant but suffers from compression loss. The raw-groundedis the most accurate but expensive. TierMem (router) closes most of the accuracy gap while using roughly half the input tokens of the raw-grounded.

TierMem (router) reaches **0.851** accuracy versus **0.873** for the raw-grounded, while reducing average input tokens by **54.1%** and latency by **60.7%**. Relative to TierMem (summary-only), the router recovers much of the loss caused by summary compression. The most plausible interpretation is exactly the one suggested by the diagnosis section: many failures are not due to memory being useless overall, but due to summaries being insufficient for specific queries.

Table 2: **Main results on LongMemEval.** Fine-grained subtype scores are in appendix A.2

System	Acc. ↑	UOR ↓	Tok$_{in}$ ↓	Lat. ↓
Mem0	0.596	29.6	1268	15.2
LightMem	0.716	16.8	1160	7.87
O-mem	0.620	25.7	2680	12.3
MemOS	0.652	27.4	1682	2.3
MemR3	0.742	–	6347	18.9
GAM	0.764	–	37850	42.3
TierMem (summary-only)	0.678	19.6	**380**	**5.37**
TierMem (raw-grounded)	**0.808**	–	7298	13.56
TierMem (router)	0.752	–	4895	10.04
TierMem (router, 4.1-mini)	0.770	–	4574	9.74

Table 3: **Root causes of summary-only errors** on LoCoMo ($n = 1540$; summary-only errors $= 378$). The top two categories are direct manifestations of the write-before-query asymmetry.

Root cause	#	%
Information missing	156	41.3
Over-generalization	109	28.8
Reasoning failure	43	11.4
Retrieval mismatch / ambiguity	20	5.3
Temporal confusion	11	2.9
Entity confusion	6	1.6
Other	33	8.7

5.2 LongMemEval

Table 2 shows the same pattern on LongMemEval. Summary-centric systems incur high UOR, which is consistent with the benchmark's emphasis on knowledge updates, temporal reasoning, and multi-session evidence integration.

The routed system underperforms the always-raw reference, as expected, but still offers a much better operating point than reading raw logs for every query. This is the systems result the paper should foreground: TierMem is not trying to prove that summaries or raw logs are globally superior. It shows that long-horizon memory benefits from *query-conditioned selective verification*.

6 Analysis

This section explains *why* TierMem improves the frontier. We focus first on failure anatomy, then on provenance links, then on write-back amortization, and only finally on router behavior.

6.1 Anatomy of summary-only failures

To understand the write-before-query asymmetry more directly, we analyze the subset of LoCoMo queries where TierMem (summary-only) fails. The resulting taxonomy in Table 3 shows that the dominant causes are **missing information** and **over-generalization**. Together they account for **70.1%** of summary-only errors.

This analysis matters for positioning. It shows that summary-only memory fails in systematic ways tied to lossy compression under unknown future queries. That is precisely the setting where selective verification should help.

6.2 Provenance links improve the escalated path

A key design choice in TierMem is that summary units point back to supporting raw pages. To isolate this mechanism, we compare **Linked** TierMem against a **No-Linked** variant that still escalates but does not prioritize linked raw pages during escalation.

Table 4: **Effect of provenance links.** Improvements are concentrated on escalated queries, not on the summary-only path.

Method	Acc.	R-rate	Acc.@R	Tok$_{in}$	Lat.
Raw-grounded reference	87.3	100%	87.3	7398	17.18
No-Linked	83.6	38.0%	77.5	3832	6.33
Linked (ours)	**85.1**	39.0%	**81.7**	3980	6.76

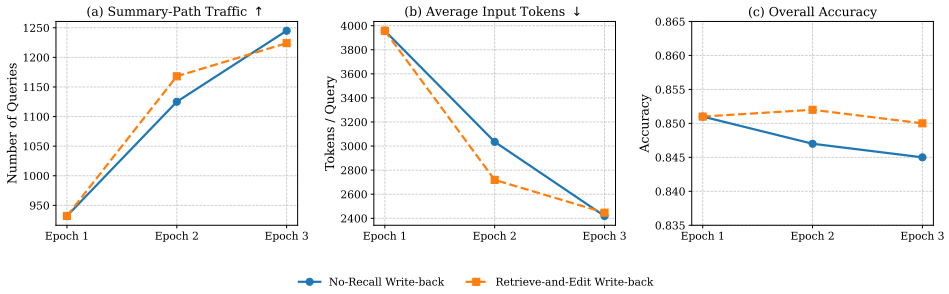

Figure 3: Write-back amortization under replay. Across replay epochs, summary-path traffic increases and average input tokens decrease while overall accuracy remains stable.

Linked TierMem improves end-to-end accuracy by **1.5** points over No-Linked, with nearly identical routing rates. The gain is concentrated on escalated queries: **Acc.@R** rises from **77.5%** to **81.7%**. This is the expected causal role of provenance links. They do not change what summaries contain; they improve how the system finds the right raw evidence once escalation happens.

Linked-evidence recovery. We also quantify a simple provenance-linked recovery signal. Among the **378** LoCoMo queries that the summary-only system answers incorrectly, **45** are corrected after the system consults the *linked* raw pages. This supports the claim that provenance links are not merely bookkeeping: they help recover answer-supporting evidence for a measurable subset of otherwise failed queries.

6.3 Verified write-back amortizes future cost

We evaluate write-back in an epoch-wise replay protocol. Within each epoch, Tier-1 is frozen while the system answers the full query set. Verified findings from escalations are logged during the epoch and written back only between epochs. This keeps evaluation conditions within an epoch fixed while revealing whether write-back reduces future reliance on the raw path.

Across replay epochs, overall accuracy remains roughly stable while the fraction of queries answered on the cheap summary path increases. Under no-recall write-back, the number of correctly answered summary-path queries rises from **819** to **1083**, while average tokens fall from **3958** to **2419**. Retrieve-and-edit write-back shows a similar trend while improving memory hygiene by updating related entries rather than only appending new ones.

The important systems takeaway is that selective verification is not only useful per query. It also improves the memory state over time by turning past escalations into future cheap access.

6.4 The router acts as a lightweight sufficiency controller

Table 5 shows that training improves the router's decision boundary, especially by increasing recall on hard queries while keeping the controller overhead modest.

Table 5: **Router behavior on LoCoMo.** The main text keeps only the compact controller story. Full training details and counterfactual analyses are moved to the appendix.

System	R-rate	Recall@hard	Acc.	Router tok.
TierMem (summary-only)	0%	–	0.755	–
TierMem (raw-grounded)	100%	100.0%	**0.873**	–
Zero-shot router	20.6%	44.7%	0.806	901
SFT router	38.2%	65.8%	0.831	680
SFT+GRPO router	39.0%	71.7%	0.851	678
GPT-4.1-mini router	35.1%	**76.4%**	0.851	685

7 Related Work

7.1 Write-time compressed memory

Long-horizon LLM agents often rely on external memory because feeding full histories into the context window is costly and can be brittle Chen et al. (2023); Liu et al. (2023). Many memory systems therefore compress interaction histories into summaries, facts, graphs, or other compact units Packer et al. (2024); Zhong et al. (2023); Chhikara et al. (2025); Fang et al. (2025); Rasmussen et al. (2025). These systems improve efficiency, but they generally treat memory quality as a write-time compression problem. TierMem instead emphasizes the downstream consequence of that choice: compressed memory must still support query-specific evidence recovery.

7.2 Adaptive retrieval and control

Recent work in adaptive retrieval shows that retrieval should depend on need rather than being invoked uniformly Jiang et al. (2023); Asai et al. (2023); Shinn et al. (2023). Closer to agent memory, systems such as MemR^3 and GAM allocate additional inference-time budget to gather stronger evidence under cost constraints Du et al. (2025); Yan et al. (2025). TierMem shares the selective-access intuition but differs in emphasis: its central object is the *memory architecture* that supports selective verification across evidence granularities, not the training method for the controller itself.

7.3 Provenance and verifiable grounding

A growing line of work studies retrieval pipelines that support revision, citation, and verifiable grounding Gao et al. (2022); Cao & Wang (2024). TierMem is most closely connected to this direction through its provenance links and write-back policy. The difference is that we study provenance in a long-horizon agent-memory setting, where compressed memory must remain connected to immutable raw evidence and where repeated verification can be amortized through provenance-preserving write-back.

8 Conclusion

This paper argues that long-horizon agent memory should be formulated as an *inference-time evidence allocation* problem, not only a write-time compression problem. Because memory is written before future queries are known, summary-only systems face a structural **write-before-query asymmetry**: compression is query-agnostic, but answer correctness depends on query-specific evidence. Always grounding on raw history avoids this failure mode, but it is too costly to be the default.

TierMem addresses this with a provenance-linked two-tier architecture: summary-first retrieval for a cheap default path, selective escalation to immutable raw evidence when summaries are insufficient, and verified write-back to amortize future cost. Across LoCoMo and LongMemEval, this design improves the accuracy–efficiency frontier relative to static summary-only memory while substantially reducing the cost of always-raw grounding.

The broader takeaway is a systems principle for long-horizon agents: because future queries are unknown at write time, useful memory must support *selective verification* against authoritative evidence rather than rely on fixed compression alone.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection, 2023. URL https://arxiv.org/abs/2310.11511.

Shuyang Cao and Lu Wang. Verifiable generation with subsentence-level fine-grained citations, 2024. URL https://arxiv.org/abs/2406.06125.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation, 2023. URL https://arxiv.org/abs/2306.15595.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Xingbo Du, Loka Li, Duzhen Zhang, and Le Song. MemR^3: Memory retrieval via reflective reasoning for LLM agents, 2025. URL https://arxiv.org/abs/2512.20237.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation, 2025. URL https://arxiv.org/abs/2510.18866.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Y. Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. RARR: Researching and revising what language models say, using language models, 2022. URL https://arxiv.org/abs/2210.08726.

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.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation, 2023. URL https://arxiv.org/abs/2305.06983.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhengren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofen Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. MemOS: A memory OS for AI system, 2025. URL https://arxiv.org/abs/2507.03724.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL https://arxiv.org/abs/2307.03172.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents, 2024. URL https://arxiv.org/abs/2402.17753.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2024. URL https://arxiv.org/abs/2310.08560.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. URL https://arxiv.org/abs/2501.13956.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Piaohong Wang, Motong Tian, Jiaxian Li, Yuan Liang, Yuqing Wang, Qianben Chen, Tiannan Wang, Zhicong Lu, Jiawei Ma, Yuchen Eleanor Jiang, and Wangchunshu Zhou. O-mem: Omni memory system for personalized, long horizon, self-evolving agents, 2025. URL https://arxiv.org/abs/2511.13593.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory, 2025. URL https://arxiv.org/abs/2410.10813.

B. Y. Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General agentic memory via deep research, 2025. URL https://arxiv.org/abs/2511.18423.

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.

A Additional Experimental Details and Full Tables

This appendix collects material that should not dominate the main paper: full subtype results, output-token statistics, detailed latency notes, and broader routing-cost breakdowns.

A.1 Full LoCoMo Results

Table 6: Full LoCoMo results with subtype breakdown, output tokens, and latency.

	Accuracy (LLM-Judge) ↑					UOR %↓	F1↑	Efficiency		
System	**Multi**	**Temp.**	**Open**	**Single**	**Overall**	**Rate**	**Overall**	**Tok_{in} ↓**	**Tok_{out} ↓**	**Lat.(s) ↓**
Mem0 Chhikara et al. (2025)	0.628	0.707	0.531	0.712	0.684	23.3	0.426	1085.9	8.0	20.32
LightMem Fang et al. (2025)	0.709	0.807	0.542	0.810	0.773	14.7	0.495	1159.5	6.9	27.69
O-mem Wang et al. (2025)	0.706	0.776	0.604	0.797	0.764	16.3	0.512	2178.1	544.2	1.16
MemOS Li et al. (2025)	0.773	0.639	0.625	0.804	0.753	17.1	0.400	2692.3	13.3	10.80
MemR3 Du et al. (2025)	0.812	0.829	0.729	0.902	0.860	–	0.586	4585.7	384.1	12.30
GAM Yan et al. (2025)	**0.858**	0.860	**0.781**	0.886	0.869	–	0.574	15773.7	1231.5	21.85
TierMem (summary-only)	0.656	0.717	0.667	0.812	0.755	15.4	0.514	**260.1**	**6.1**	**1.24**
TierMem (raw-grounded)	0.812	**0.863**	0.719	**0.916**	**0.873**	–	**0.604**	7398.4	521.3	17.18
TierMem (router)	0.794	0.829	0.708	0.895	0.851	–	0.581	3980.4	341.5	6.76
TierMem (router, GPT-4.1-mini)	0.755	0.850	0.677	0.904	0.851	–	0.591	3620.9	235.4	6.40

A.2 Full LongMemEval Results

Table 7: Full LongMemEval results with subtype breakdowns.

Method	**SS-User**	**SS-Asst**	**SS-Pref**	**Multi-S**	**Know. Upd**	**Temp. Reas**	**Overall**	**UOR %↓**	**Tok_{in}/Q**	**Tok_{out}/Q**	**Lat./Q**
Mem0	0.800	0.589	0.533	0.571	0.667	0.489	0.596	29.6	1268	9	15.2
LightMem	0.871	0.339	0.767	**0.790**	**0.885**	**0.700**	0.716	16.8	1160	7	7.87
O-mem	0.957	0.446	0.773	0.632	0.551	0.542	0.620	25.7	2680	670	12.3
MemOS	0.872	0.617	0.963	0.514	0.840	0.557	0.652	27.4	1682	206	2.3
MemR3	0.929	0.929	0.600	0.684	0.769	0.639	0.742	–	6347	437	18.9
GAM	0.957	0.964	0.600	0.752	0.756	0.629	0.764	–	37850	1448	42.3
TierMem (summary-only)	0.900	0.857	0.833	0.549	0.756	0.519	0.678	19.6	**380**	**11**	**5.37**
TierMem (raw-grounded)	**0.971**	**0.982**	**0.900**	0.752	0.846	0.662	**0.808**	–	7298	677	13.56
TierMem (router)	0.957	0.946	0.833	0.692	0.744	0.602	0.752	–	4895	455	10.04
TierMem (router, GPT-4.1-mini)	0.929	**0.982**	0.867	0.714	0.782	0.624	0.770	–	4574	328	9.74

B Implementation Details

B.1 Memory Instantiation

Our framework is general; in experiments we instantiate the memory management system using the open-source **Mem0** library Chhikara et al. (2025) for vector indexing and storage. We adapt Mem0 to our **Paged-Log Architecture** as follows:

- **Summarization prompt (f_{summ}).** We replace Mem0's default entity-graph extraction with **page-level summarization**. The prompt instructs the model to generate a concise, self-contained summary of the interaction log, preserving key dates, decisions, and user preferences.
- **Storage schema.** Each page summary is embedded and indexed in the vector store. We extend Mem0 metadata to include the Page ID (P_i) as a provenance pointer ρ_i, enabling retrieval to map back from a summary vector to its originating page.
- **Embedding model.** We embed summaries with `text-embedding-3-small`.

Mem0 components used vs. disabled. In our implementation, Mem0 is used only for (i) storing and querying embeddings for Tier-1 summary entries and (ii) attaching/retrieving metadata (e.g., `raw_log_id/page_id`) for provenance. We do not use Mem0's entity-graph construction, graph updates, or graph-based retrieval; these components are disabled/bypassed so that Tier-1 consists solely of page summaries plus metadata pointers.

B.2 PAGED-LOG INGESTION AND PERSISTENCE

We implement a **paged ingestion** procedure that converts streaming dialogue turns into fixed-size *pages* stored in a persistent `PageStore`. Each incoming turn is appended to the current page with an explicit timestamp and speaker tag. Once a page reaches a configured maximum length (default: 1000 tokens), we trigger a summarization-and-indexing step via `mem0.add(infer=True)` over the entire page.

Page format. Each page stores raw interaction text line-by-line in the canonical form:

$$[\text{timestamp}]\ \text{speaker}:\ \text{text}.$$

This preserves multi-speaker context for datasets with more than one interlocutor.

Provenance. For each page P_i, we generate a unique `page_id`. We set `raw_log_id` $\leftarrow$ `page_id` in Mem0 metadata such that any retrieved summary hit can be deterministically mapped back to the original raw page content in `PageStore`.

Durability and replay. `PageStore` persists pages to disk and supports session-scoped loading. This enables resuming runs without re-ingesting all prior turns and ensures that the raw evidence used for research-mode grounding remains accessible.

B.3 DEFERRED SUMMARIZATION FOR UNINDEXED PAGES

To handle sessions that terminate before a page fills (or failures during ingestion), we provide a **deferred summarization** routine `auto_summary`. It enumerates pages whose summaries have not yet been stored and calls `mem0.add(infer=True)` for each such page. The returned memory strings are recorded in `PageStore` and the page is marked as indexed, ensuring that the vector store remains consistent with the persistent log.

B.4 QUERY ANSWERING PIPELINE

Given a query q, we first retrieve top-k summary hits from Mem0 using vector similarity search. Each hit contains a summary text and a provenance pointer to its source page.

Two-path routing (S/R). We use a lightweight router to select between:

- **S-path (Summary-only).** When retrieved summaries contain an explicit answer, we generate the response directly from the summaries.
- **R-path (Research).** When summaries are ambiguous or incomplete, we switch to a slower research mode that grounds generation in raw page evidence.

Cost accounting. We record (i) Mem0 retrieval token usage and latency, (ii) router token usage, and (iii) answer-generation token usage. These are aggregated into online cost metrics for analysis.

B.5 RESEARCH MODE: INTEGRATION–PLAN LOOP

The R-path implements an iterative loop that alternates between **integration** (extracting grounded facts from evidence) and **planning** (deciding whether to retrieve more evidence).

Evidence construction. We group retrieved hits by `page_id` and build an evidence block that includes: (i) Mem0 page summaries (when available) and (ii) the full raw page content from `PageStore`. This ensures that research mode always has access to verbatim text for grounding.

Integration step. The integration prompt extracts a set of *linked facts* of the form:

$$\texttt{fact},\ \texttt{evidence\_quote}.$$

Crucially, we *do not* ask the model to generate provenance identifiers.

Deterministic provenance linking. After integration, we map each extracted fact to its source pages using a deterministic post-processing step:

- First, we attempt substring matching of the normalized `evidence_quote` against each page's raw content (and, if needed, its summaries).
- If quote matching fails, we fall back to keyword overlap between the fact text and page content.

This produces `source_pages` and `evidence_snippets` without relying on model-generated page identifiers, reducing hallucinated provenance.

Planning step. A separate planning prompt evaluates coverage and returns either `DONE` or `SEARCH` with retrieval commands. We support two retrieval operators:

- **MEM0_SEARCH**: semantic retrieval over Mem0 summaries.
- **KEYWORD_SEARCH**: BM25-style keyword search over raw pages in `PageStore`.

We deduplicate issued commands and terminate early when no new pages are discovered.

B.6 Dynamic Write-back Protocol

We implement online consolidation as a closed-loop memory update step. Given a set of *verified* linked facts extracted during the Research Integration phase:

1. Conflict Retrieval. For each new candidate fact f_{new}, we query $\mathcal{M}_S$ to retrieve the top-k (e.g., $k = 3$) most similar existing summary units, denoted as $\mathcal{M}_{retrieved}$.

2. Operation Selection. We prompt an LLM (the *Memory Manager*) to compare f_{new} with $\mathcal{M}_{retrieved}$. The model outputs one of three actions:

- `SKIP`: If f_{new} is semantically entailed by $\mathcal{M}_{retrieved}$.
- `UPDATE`: If f_{new} adds specific details (e.g., timestamps, names) to a vague existing entry. The model generates the merged text.
- `ADD`: If f_{new} represents a distinct, previously unindexed event.

3. Provenance Inheritance. If an `UPDATE` occurs, the new unit inherits the provenance pointers ρ from both the original unit and the new evidence sources, maintaining a complete audit trail.

B.7 Reranking (Optional)

When enabled, we apply a **page-level reranker** to candidate evidence pages. Hits are first grouped by `page_id`; reranking operates on each page's raw content (or, if missing, concatenated summaries) and selects the top-k *unique pages*. We also include a **protection rule**: pages with high-confidence Mem0 similarity scores are retained to avoid discarding highly relevant memories before reranking.

Discussion: Online Consolidation vs. Epoch-wise Replay Evaluation TierMem is designed to support *online* consolidation: after an escalation, verified findings can be written back immediately to reduce future misses. However, to match the benchmark evaluation protocol and avoid within-run distribution shift, our main experiments report results under an epoch-wise replay setting where Tier-1 is held fixed within each epoch and updates are applied between epochs. We expect that in interactive deployments, moving write-back into the QA loop is straightforward and may further reduce repeated escalations, but careful design is needed to preserve reproducibility and to avoid unintended interactions with evaluation sampling.

Algorithm 1 TierMem QA (Abstract): Router + Bounded Retrieve–Integrate–Plan Loop

Require: Query Q; Tier-1 summaries $\mathcal{M}_S$; Tier-2 raw pages $\mathcal{M}_R$
Require: Router π_θ; generator Gen
Require: Retrieve, Rerank, Integrate, Plan
Require: max iterations $T_{\max}$
Ensure: Answer $\hat{Y}$

$Z_0 \leftarrow \mathrm{Retrieve}(\mathcal{M}_S, Q)$
if $\pi_\theta(Q, Z_0) = \textsc{Answer}$ **then**
  return $\mathrm{Gen}(Q, Z_0)$
end if
$E \leftarrow \mathrm{Integrate}(Q, \mathrm{Linked}(Z_0))$ ▷ seed: summaries + linked raw pages (compressed)
$q_1 \leftarrow Q$; $H \leftarrow \emptyset$ ▷ optional: searched queries / history
for $t = 1$ **to** $T_{\max}$ **do**
  $\mathcal{C}_t \leftarrow \mathrm{Retrieve}(\mathcal{M}_S, q_t) \cup \mathrm{Retrieve}(\mathcal{M}_R, q_t)$
  $X_t \leftarrow \mathrm{TopK}\big(\mathrm{Rerank}(\mathcal{C}_t, Q)\big)$
  $E \leftarrow E \cup \mathrm{Integrate}(Q, X_t)$
  $(d_t, q_{t+1}) \leftarrow \mathrm{Plan}(Q, E, H)$ ▷ $d_t \in \{\textsc{Stop}, \textsc{Search}\}$
  if $d_t = \textsc{Stop}$ **then**
    break
  end if
  $H \leftarrow H \cup \{q_t\}$; $q_t \leftarrow q_{t+1}$
end for
$\hat{Y} \leftarrow \mathrm{Gen}(Q, E)$
▷ optional: $\mathrm{WriteBack}(\mathcal{M}_S, E)$
return $\hat{Y}$

B.8 Statistics and Debug Instrumentation

For analysis, we log per-query events including (i) route choice (S vs. R), (ii) hit breakdown by `source_type` (`original` vs. `linked_fact`), (iii) linked-fact write-back counts, and (iv) latency/token usage. This instrumentation is used only for evaluation and does not affect the core algorithm.

C Write-Back Replay Table

Table 8: **System evolution under consolidation on LoCoMo ($n = 1540$).** This full table is moved to the appendix; the main paper should visualize the trends.

Epoch	Overall Acc.	S-Traffic	S-Acc.	S-Correct ↑	$\mathrm{Tok}_{\mathrm{avg}}$	Lat.(s)	ADD	UPDATE
No-recall write-back (only ADD or SKIP)								
E1	0.851	932	87.9%	819	3958	5.14	720	–
E2	0.847	1125	87.6%	985 (+166)	3035	4.16	486	–
E3	0.845	1245	87.0%	1083 (+98)	2419	3.39	317	–
Retrieve-and-edit write-back								
E1U	0.851	932	87.9%	819	3958	5.14	1186	209
E2U	0.852	1168	87.0%	1016 (+197)	2719	4.22	652	105
E3U	0.850	1224	86.8%	1063 (+57)	2447	3.85	426	77

D Router: Data Construction, Training Pipeline, and Hyperparameters

This appendix consolidates all details related to router supervision, SFT distillation, GRPO optimization, and the exact hyperparameters used.

D.1 TASK DEFINITION AND NOTATION

The router is a lightweight controller that decides whether the system can answer a user query using Tier-1 summaries alone (ANSWER/S) or must escalate to Tier-2 raw logs (ESCALATE/R). Given a query Q and an initial evidence state Z_0 consisting of the top-k retrieved Tier-1 summaries, the router predicts:

$$a \sim \pi_\theta(a \mid Q, Z_0), \qquad a \in \{\texttt{S}, \texttt{R}\}.$$

The router output is *strictly* a JSON object with an action token (and optionally a teacher `thinking` field during distillation).

D.2 DATA CONSTRUCTION: SUMMARY SUFFICIENCY LABELS

Source benchmark slice. We construct router training data from the *Long-Range Understanding (LRU)* competency in MemoryAgentBench Hu et al. (2025), using a summarization-style subset of 2,000 instances. Each instance is materialized into a linked two-tier memory state $(\mathcal{M}_S, \mathcal{M}_R)$ using the write path described in §3.1. For each query Q, Tier-1 dense retrieval returns the initial summary evidence Z_0 (top-k_s summaries), which forms the router input.

Two-path execution (Summary vs. Research). For each query Q, we execute two fixed policies to determine whether Z_0 is sufficient:

- **Summary-only (S-path):** $\hat{Y}_S \leftarrow \mathrm{Gen}(Q, Z_0)$
- **Research/raw-grounded(R-path):** Run the bounded Deep Search Loop (§3.2) to obtain E_{final}, then $\hat{Y}_R \leftarrow \mathrm{Gen}(Q, E_{\text{final}})$

LLM judge and sufficiency rubric. We obtain binary indicators $c_S, c_R \in \{0, 1\}$ using LLM-based judges with conservative rubrics. Both judges are run with temperature 0 and are constrained to output a fixed JSON schema to reduce variance.

S-path: summary sufficiency judge. Given $(Q, y^\star, Z_0)$, the judge determines whether the retrieved summaries contain *explicitly sufficient* information to answer the question correctly *without guessing*. We use the following prompt (verbatim):

D.3 JUDGE PROMPT FOR SUMMARY SUFFICIENCY

We set $c_S = \mathbb{I}[\texttt{has\_sufficient\_info} = \texttt{true}]$.

R-path: research/correctness judge. Given $(Q, y^\star, \hat{Y}_R)$, the judge determines whether the research-path answer matches the gold answer. We use a separate LLM-as-judge prompt (Appendix D.3), which outputs a JSON object of the form: {"label": ``CORRECT'' / ``WRONG''}. We set $c_R = \mathbb{I}[\texttt{label} = \text{``CORRECT''}]$.

Why conservative judging? Our routing label targets *summary sufficiency*. False positives (marking summaries as sufficient when they are not) would train a router that under-escalates and produces ungrounded answers. Therefore, the S-path prompt intentionally biases toward `false` when there is ambiguity or missing explicit support.

Routing labels (semantic cache hit/miss). We define the router supervision label as:

$$y = \begin{cases} \texttt{S} & \text{if } c_S = 1, \\ \texttt{R} & \text{if } c_S = 0 \wedge c_R = 1. \end{cases}$$

We drop instances where both policies fail ($c_S{=}0 \wedge c_R{=}0$), since they provide no signal on whether escalation can recover missing evidence.

Judge Prompt (Summary Sufficiency)

You are evaluating whether retrieved summaries contain sufficient information to answer a question.
Question: {question}
Gold answer: {gold_answer}
Retrieved summaries:
{summaries_text}
Your task: Determine if the summaries contain EXPLICIT information to answer the question correctly.
Strict criteria (be conservative):

1. The answer should be DIRECTLY stated or clearly derivable from the summaries.
2. Do NOT count vague/related information as sufficient.
3. Do NOT infer causes from effects.
4. For completeness questions ("how many", "list all", "both"), summaries must be COMPLETE.
5. For exact details (dates, numbers, names), summaries must contain those exact details.

If you have ANY doubt whether the summaries allow a factual answer without guessing, answer false.
Output JSON:

```
{
  "has_sufficient_info": true/false,
  "reason": "..."
}
```

Figure 4: The judge prompt used to evaluate whether the retrieved summaries provide sufficient context to answer the ground truth.

D.4 Training Workflow Overview

We train the router using a two-stage pipeline:

1. **Stage 1 (SFT / Distillation):** Initialize the router to follow the routing protocol and learn strong sufficiency heuristics from a high-capacity teacher (GPT-5) using a strict JSON format.
2. **Stage 2 (GRPO):** Starting from the SFT checkpoint, refine the decision boundary under an explicit accuracy–cost trade-off, discouraging unnecessary escalation while preserving faithfulness.

This separation is intentional: SFT primarily regularizes behavior and format; GRPO optimizes the accuracy–efficiency trade-off.

D.5 Stage 1: SFT via GPT-5 Thinking+Action Distillation

Teacher prompt (GPT-5 thinking trace + binary action). We distill a teacher-generated `thinking` trace and the final routing decision `action` using the following prompt (verbatim):

D.6 Router Thinking Prompt

SFT targets and formatting. For each example, the student router is trained to reproduce the teacher's JSON response. We treat the entire JSON (including `thinking` and `action`) as the supervised target, using standard next-token cross-entropy. At inference time, the system uses only the `action` field.

Teacher-label consistency filter. To ensure alignment with oracle labels derived in §D.2, we keep only SFT examples where the teacher action matches the oracle routing label y. Examples with mismatched teacher actions are discarded.

Router Thinking Prompt

You are a router for a memory-augmented QA system. Decide whether the retrieved summaries are sufficient to answer the question.

Available Actions

1. **"S" — Answer using current summaries only**
   Use when: The summaries contain a span that directly states the answer (or a direct paraphrase) WITHOUT needing to assume missing facts.
   Rules of thumb:
   - If you can point to a specific snippet that can be copied into the answer, choose S.
2. **"R" — Deep research mode**
   Use when: Summaries are too ambiguous, contradictory, or unrelated; OR the question requires a comprehensive list and summaries are clearly incomplete.
   R is more reliable than S.

Input

QUESTION: {question}
SUMMARIES:
{summaries_block}

Decision checklist

- What exact answer format is needed? (single value / list / date / identity)
- Do summaries explicitly contain it?
- Any ambiguity or need to infer?
- Any completeness requirement?

Output (JSON only)

```
{
  "thinking": "<your reasoning>",
  "action": "S/R"
}
```

Figure 5: The updated router prompt incorporating explicit chain-of-thought reasoning to improve decision reliability.

Why include thinking in SFT. The teacher `thinking` field provides structured intermediate reasoning about evidence sufficiency (e.g., completeness, ambiguity, exactness), which improves stability and reduces instruction-following failures (e.g., verbose or malformed outputs) in a small router.

D.7 Stage 2: GRPO Alignment for Accuracy–Cost Trade-off

Starting from the SFT checkpoint, we optimize the router with GRPO using a trajectory-level reward:

$$R(\tau) = R_{\text{acc}}(\tau) \; - \; \lambda_{\text{cost}} R_{\text{cost}}(\tau) \; - \; \lambda_{\text{waste}} R_{\text{waste}}(\tau), \tag{1}$$

where:

- R_{acc} encourages correct routing (choose S when summaries suffice; choose R when escalation is necessary),
- R_{cost} penalizes choosing R to reflect token/latency overhead,
- R_{waste} penalizes *unnecessary escalation* (choosing R when $y = \texttt{S}$).

Concrete reward implementation. In our implementation, rewards are computed from (c_S, c_R) and the router decision a:

GRPO dataset construction. We build GRPO prompts using the same router input (Q, Z_0) as in SFT. For each prompt, we sample multiple router completions per prompt (generations per prompt is reported in §D.9) to compute group-relative advantages.

D.8 GRPO Dataset Preparation and Filtering

Starting from the offline QA logs (which include both summary-path and research-path outcomes), we apply the following filters:

1. Remove instances where **both** paths fail ($c_S = 0 \wedge c_R = 0$).

2. Remove instances where the **R-path fails** ($c_R = 0$), since escalation does not recover evidence.

3. Optionally re-judge S-path sufficiency using a fixed judge configuration to reduce label noise; keep only instances consistent under re-judging.

Optional augmentation. We optionally augment a subset of questions by paraphrasing while preserving intent and specificity (2–3 paraphrases per question). Paraphrases inherit the same memory state and labels.

Balancing. To emphasize hard decisions, we oversample cases where escalation is necessary ($c_S = 0 \wedge c_R = 1$) during GRPO minibatch formation.

D.9 Hyperparameters

D.9.1 SFT Hyperparameters

Parameter	Value
Train type	LoRA
Precision	bfloat16
Epochs	5
Learning rate	1×10^{-4}
Max sequence length	4096
Per-device train batch size	8
Per-device eval batch size	4
Gradient accumulation steps	1
LoRA rank / alpha	64 / 128
Target modules	all-linear
Warmup ratio	0.1
Eval / Save steps	10 / 10
Save total limit	5
Logging steps	1
Dataloader workers	4
Loss scaling	ignore_empty_think
Kernel optimization	use_liger_kernel=true
DeepSpeed	ZeRO-2

Table 9: SFT (LoRA) training configuration for the router.

D.9.2 GRPO HYPERPARAMETERS

Parameter	Value
RLHF type	GRPO
Initialization	SFT adapter checkpoint
Precision	bfloat16
Epochs	6
Context length	2048
Learning rate	5e-6
Max completion length	256
Per-device train batch size	24
Per-device eval batch size	24
Gradient accumulation steps	1
Warmup ratio	0.05
DeepSpeed	ZeRO-2
Target modules	all-linear
LoRA rank / alpha	64 / 128
Dataloader workers	4
Logging / Save steps	5 / 50
Completions logging	log_completions=true
Sampling temperature	1.2
Generations per prompt	8
Rollout forward batch size	8
KL / regularization parameter	$\beta = 0.02$
Reporting	Weights & Biases

Table 10: GRPO training configuration for router policy optimization.

D.9.3 REWARD COEFFICIENTS

Name	Value
Correct reward (`CORRECT_REWARD`)	1.0
Wrong penalty (`WRONG_PENALTY`)	-1.5
Cost of S (`COST_S`)	0.0
Cost of R (`COST_R`)	0.1
Waste penalty for unnecessary R (`WASTE_R`)	0.4
Format error penalty (`FORMAT_ERROR_PENALTY`)	-1.0

Table 11: Reward coefficients used by the router GRPO objective.

E DETAILED ANALYSIS OF PROVENANCE POINTER ABLATION

This appendix provides a detailed analysis of the provenance pointer ablation study introduced in Section 6.2. We analyze route consistency, accuracy under identical routing decisions, and token-level cost, in order to isolate the effect of provenance-aware warm-start from routing behavior or additional computation.

E.1 ROUTE CONSISTENCY

Across 1,540 shared queries, Linked and No-Linked make identical routing decisions on 1,306 queries (84.8%), and differ on 234 queries (15.2%). This indicates that enabling provenance pointers induces only modest changes in routing behavior. Therefore, improvements observed in the Linked setting cannot be explained solely by more aggressive escalation, but instead suggest higher-quality evidence retrieval during escalation.

E.2 Accuracy under Identical Routes

To disentangle retrieval quality from routing effects, we analyze queries where Linked and No-Linked select the same route.

Both Escalate (R). When both systems escalate to raw logs, Linked answers 31 queries correctly that No-Linked answers incorrectly, whereas the reverse occurs in 25 cases. This yields a net gain of +6 queries for Linked under identical escalation decisions.

Both Answer (S). When both systems answer directly from summaries, Linked answers 16 queries correctly that No-Linked answers incorrectly, compared to 10 in the opposite direction. The small difference confirms that provenance pointers primarily affect raw evidence grounding rather than summary-only answering.

These controlled comparisons demonstrate that provenance pointers improve the *quality* of retrieved evidence, independent of routing frequency.

E.3 Cost Analysis by Route

Table 12 reports average token usage by route. On summary-only queries, Linked and No-Linked incur nearly identical cost. On escalated queries, Linked incurs slightly higher token usage, reflecting deeper or more precise raw evidence grounding.

Importantly, the average research depth is nearly identical between Linked (1.38) and No-Linked (1.36), with the same maximum depth (3), indicating that gains do not stem from additional retrieval iterations.

Method	Route	Tok_{in}	Tok_{out}
Linked	S	847.5	97.8
Linked	R	8875.1	724.0
No-Linked	S	847.2	96.8
No-Linked	R	8705.0	709.9

Table 12: Average QA token cost by route for provenance pointer ablation.

E.4 Summary

In summary, provenance pointers consistently improve performance on escalated queries by enabling more reliable raw-log grounding. The improvement persists under identical routing decisions and does not rely on increased search depth. While provenance-aware escalation incurs slightly higher token cost, it yields a favorable accuracy–efficiency trade-off, validating provenance linking as a core component of TierMem.

F Additional Analysis: Why Higher Hard-Recall Can Tie on End-to-End Judge Score

This appendix explains why GPT-4.1-mini achieves higher recall on oracle-hard cases (Table 5), yet matches our SFT+GRPO router on the final LoCoMo LLM-as-judge accuracy. Crucially, this effect does *not* come from different stored memories: all variants share identical writes and use the same memory library, generator, embedding model, and reranker (cf. evaluation protocol in §4.2). Instead, the tie arises from how routing interacts with (i) the *difficulty distribution* of escalated queries and (ii) the *effective repair/regression behavior* of the R-path under the realized evidence trajectories.

Category	Count	Decomposition (actual)
S_correct_R_correct	240	225 correct / 15 wrong
S_correct_R_wrong	33	13 correct / 20 wrong
S_wrong_R_correct	181	169 correct / 12 wrong
S_wrong_R_wrong	86	14 correct / 72 wrong
Total matched (R set)	540	421 correct / 119 wrong

Table 13: GPT-4.1-mini: S/R counterfactual comparison on the queries routed to R.

Category	Count	Decomposition (actual)
S_correct_R_correct	313	305 correct / 8 wrong
S_correct_R_wrong	26	13 correct / 13 wrong
S_wrong_R_correct	170	153 correct / 17 wrong
S_wrong_R_wrong	92	20 correct / 72 wrong
Total matched (R set)	601	491 correct / 110 wrong

Table 14: Ours (SFT+GRPO): S/R counterfactual comparison on the queries routed to R.

F.1 S/R COUNTERFACTUAL COMPARISON MATRICES ON THE R-ROUTED SET

For each router, we log a counterfactual comparison on the subset of queries routed to R: we run the S-path answer and the R-path answer and record whether each is correct under the benchmark judge. This yields a 2×2 table with counts:

- **S_correct_R_correct**: both paths would be correct.
- **S_correct_R_wrong**: R would *hurt* (regression if escalated).
- **S_wrong_R_correct**: R would *fix* S (repair benefit).
- **S_wrong_R_wrong**: both fail (hard negatives).

We report both (i) the raw S/R agreement counts and (ii) their decomposition against the final judge labels (`actual_correct/wrong`) to expose how "hard" the routed set is.

GPT-4.1-mini (R-routed subset).

Ours (SFT+GRPO) (R-routed subset).

F.2 REPAIR BENEFIT VS. REGRESSION RISK

Hard-recall is a mechanism metric that measures whether oracle-hard cases are escalated, but end-to-end judge accuracy depends on whether escalation *actually repairs* S failures without introducing regressions. From Tables 13–14, we define two diagnostics on the R-routed subset:

$$\text{RepairRate} = \Pr(R \text{ correct} \mid S \text{ wrong}) = \frac{\text{S\_wrong\_R\_correct}}{\text{S\_wrong\_R\_correct} + \text{S\_wrong\_R\_wrong}}, \tag{2}$$

$$\text{RegressRate} = \Pr(R \text{ wrong} \mid S \text{ correct}) = \frac{\text{S\_correct\_R\_wrong}}{\text{S\_correct\_R\_wrong} + \text{S\_correct\_R\_correct}}. \tag{3}$$

Measured values. For GPT-4.1-mini: $\text{RepairRate} = 181/(181+86) = 67.8\%$ and $\text{RegressRate} = 33/(33+240) = 12.1\%$. For ours: $\text{RepairRate} = 170/(170+92) = 64.9\%$ and $\text{RegressRate} = 26/(26+313) = 7.7\%$. Thus, GPT-4.1-mini shows higher repair rate but also higher regression exposure on the set it escalates, whereas our router is slightly less aggressive in "fixing" but produces fewer cases where R would be worse than S.

Table 15: Qualitative case studies illustrating how TierMem aligns retrieval granularity with query requirements.

User Query	Tier-1 Summary (Compressed)	Router Decision & Reasoning
Case 1: Mitigating Entity Loss (The "Hero" Case)		
Q: "What book recommendations has Joanna given to Nate?"	*Context:* "...Joanna recommended Nate to find a **fantasy book series**... Joanna approved the **specific book series** Nate showed her..." **✗Defect:** The specific entity (Book Title) is lost during summarization.	**Decision:** ESCALATE (Raw Search) **Router Thought:** "Summaries mention Joanna recommended a series... but **do not explicitly state the book titles**. They are contextually related but lack specific answers." **Outcome:** Retrieved raw log containing "*Little Women*", avoiding a vague or hallucinated answer.
Case 2: Semantic Sufficiency (The "Saver" Case)		
Q: "What important values does John want to teach his kids through adopting a rescue dog?"	*Context:* "...John said adopting a rescue dog... teaches John's kids **responsibility and compassion**..." **✓Sufficient:** Key semantic concepts are present.	**Decision:** ANSWER (Summary Only) **Router Thought:** "Summaries do not explicitly state 'values: X', but they hint at key points... **Since summaries contain the answer** (responsibility and compassion), use Summary." **Outcome:** Answered correctly (Accuracy preserved) while skipping ∼4k tokens of raw retrieval.

F.3 WHY END-TO-END ACCURACY CAN TIE DESPITE RECALL DIFFERENCES

Even with identical writes and identical components, routers induce different *escalated sets* and therefore different difficulty distributions for the R-path. This is visible in the **hard-negative** bucket `S_wrong_R_wrong`: GPT-4.1-mini escalates 86 such cases, while ours escalates 92. Importantly, within this bucket the judge decomposition shows that some instances labeled as `S_wrong_R_wrong` still end up as `actual_correct` (14 for GPT-4.1-mini vs. 20 for ours), which we interpret as evidence of *trajectory sensitivity* in the R-path: small differences in retrieved evidence traces and generation can flip outcomes on extremely difficult instances. In our runs, the R-path succeeds on more of these hard negatives (20 vs. 14), partially compensating for the lower oracle-hard recall.

Takeaway. Under a fixed write state, the gap between mechanism metrics (e.g., oracle-hard recall) and outcome metrics (judge accuracy) can be explained by (i) different escalated-set difficulty, (ii) repair vs. regression trade-offs, and (iii) trajectory sensitivity of the R-path on borderline hard instances. Therefore, we report not only hard recall but also the S/R counterfactual matrices to make the cost–fidelity behavior transparent.

G CASE STUDY: QUALITATIVE CASES

We present two representative cases illustrating how TierMem aligns retrieval granularity with query requirements in Table 15. Case 1 requires escalation due to entity loss in summarization, and Case 2 can be answered cheaply from summaries because key semantic evidence is preserved. A broader clustering-based analysis of router decision errors is provided in Section H.

Mitigating compression loss. When the user query requires a specific entity that is missing from Tier-1 summaries, TierMem escalates and grounds the answer in Tier-2 raw logs, avoiding vague or hallucinated responses.

Ensuring semantic efficiency. When Tier-1 summaries contain semantically sufficient evidence (even if not phrased in the same surface form as the query), the router answers directly, avoiding unnecessary raw retrieval while preserving correctness.

Error Category	Count
S→R Errors (Summary insufficient, escalation needed)	
Information completeness error	26
Disambiguation failure	17
Precision requirement ignored	16
Context and focus confusion	1
Other	1
Total S→R	**61**
R→S Errors (Unnecessary escalation)	
False information gap detection	289
Neglect of indirect or inferential evidence	49
Temporal or logical inference failure	33
Total R→S	**371**

Table 16: **Router error analysis.** We categorize routing errors into false cache hits (S→R) and false cache misses (R→S). False cache hits are relatively rare but stem mainly from summary incompleteness and unmet precision requirements. False cache misses dominate and primarily reflect conservative over-escalation, incurring additional cost without affecting correctness.

H Router Error Analysis and Failure Modes

We analyze routing errors relative, clustered into coherent categories (Table 16). We distinguish two asymmetric error types: **(i) S→R errors** (false cache hits), where the router answers from Tier-1 but raw grounding is required; and **(ii) R→S errors** (false cache misses), where the router escalates unnecessarily despite sufficient summary evidence.

False cache hits are rarer but more harmful. S→R errors are relatively rare (61 cases) but directly impact correctness. They stem mainly from *information completeness* failures (missing a required slot such as an entity name) and *precision requirement* failures (ignoring that the question demands exactness or attribution). These cases align with known brittleness of lossy compression: a summary can be topically relevant yet underspecified for exact answering.

False cache misses dominate and reflect conservative bias. R→S errors are much more frequent (371 cases). The primary cause is *false information gap detection*: the router assumes summaries are insufficient even when they contain enough evidence. These errors increase cost but typically do not degrade correctness, reflecting a conservative policy that prioritizes faithfulness. This suggests a clear path for future improvement: better calibration and sufficiency detection that leverages indirect but adequate evidence while still avoiding false cache hits.

I Prompt Templates

Fact Extraction Prompt (JSON Schema)

You are the **Universal Memory Encoder**. Your goal is to convert raw input stream into high-fidelity, self-contained knowledge records (Long-term Memory).
INPUT FORMATTING NOTICE:
The input will be provided in the next user message, prefixed by 'Input:' and often formatted as dialogue lines.
It may follow a pattern like: '[Timestamp] Speaker_Name: Content'.

- If this pattern is present, you MUST use the 'Timestamp' for temporal grounding and 'Speaker_Name' for entity resolution.
- If this pattern is absent (e.g., raw document text), treat the input as a factual source and extract knowledge propositions.

CORE OBJECTIVES:

1. **ENTITY & CONTEXT RESOLUTION**:
   - **No ambiguous references**: Every extracted fact must explicitly name the involved people, organizations, objects, places, and concepts.
   - **No "User" ambiguity**: If the input says '[...] Melanie: I like art', the fact MUST be "Melanie likes art", NOT "The user likes art".
   - **De-contextualization**: Each extracted fact must be **standalone**.
2. **TEMPORAL GROUNDING**: Use the provided timestamps in the input as the source of truth. Convert relative time ("tomorrow", "next week") into absolute context.
3. **PRONOUN & DEIXIS ELIMINATION**: Extracted facts must **not contain any pronouns**. If a pronoun appears, resolve it to the specific entity.
4. **SCENE TAGGING (EMBEDDED)**: Identify the immediate scene/activity (2-5 words). Append this scene to the end of the fact string inside brackets.

OUTPUT INSTRUCTIONS (MUST FOLLOW):
Return a JSON object with EXACTLY this schema:

```
{
  "facts": [
      "<atomic fact sentence 1> [Scene: ...]",
      "<atomic fact sentence 2> [Scene: ...]", ...
  ]
}
```

Rules: Output JSON only (no extra text, no markdown code fences).

Figure 6: The prompt used for extracting atomic facts from conversation streams.

Research Integration Prompt

You are a research assistant extracting facts from conversation evidence to answer a question.
QUESTION: {question}
RETRIEVED EVIDENCE: {evidence}
YOUR TASK:
Read through ALL pages in the evidence and extract facts that help answer the question.
EXTRACTION STRATEGY:
Identify what information is needed and extract:

1. **Direct answers**: Facts that directly answer the question
2. **Component facts**: Facts about entities/topics in the question that can be combined
3. **Temporal facts**: When events happened
4. **Confirmation facts**: Look for acceptance/approval messages

OUTPUT FORMAT (JSON):

```
{
  "linked_facts": [
    {
      "fact": "<extracted fact - use specific dates/names>",
      "evidence_quote": "<EXACT quote from the evidence>"
    }
  ],
  "coverage_assessment": "<what aspects are covered vs missing>"
}
```

Figure 7: Prompt for integrating retrieved facts to answer a query.

Research Plan Prompt

You are a research planner. Based on the current facts and the question, decide if more information is needed.
QUESTION: {question}
CURRENT FACTS: {current_facts}
COVERAGE ASSESSMENT: {coverage_assessment}
RESEARCH HISTORY: {research_history}
ALREADY SEARCHED: {searched_queries}
YOUR TASK:
Decide whether the current facts are SUFFICIENT to answer the question, or if more search is needed.
OUTPUT FORMAT (JSON):

```
{
  "decision": "DONE" or "SEARCH",
  "reasoning": "<brief explanation>",
  "search_commands": [
    {"type": "MEM0_SEARCH", "query": "<semantic search query>"},
    {"type": "KEYWORD_SEARCH", "keywords": ["k1", "k2"]}
  ]
}
```

DECISION CRITERIA:

- **DONE**: The facts contain enough information to provide a reasonable answer.
- **SEARCH**: Key information is missing and more search might help.

Figure 8: Prompt for planning the next research step or concluding the search.

Router Prompt (Summarization vs. Research Decision)

You are an expert router for a memory-augmented QA system. Analyze the retrieved summaries and decide the best action to answer the question.
Available actions:
1. "S" - Answer using current summaries only. Use when Summaries contain the EXPLICIT answer.
2. "R" - Deep research mode (slow path). Use when Summaries are ambiguous, only contextually related, or miss the answer entirely.
Question: {q}
Retrieved Summaries: {summaries_block}
Output format (JSON only):

```
- If answering with summaries: {"action": "S"}
- If deep research needed: {"action": "R"}
```

Figure 9: Routing prompt to decide between immediate summarization or deep research.

Answer Generation Prompts (Template Functions)

```
def _make_locomo_summary_prompt(summary: str, question: str) -> str:
    return f"""\
Based on the summary below, write an answer in the form of **a short phrase**...
Answer with exact words from the context whenever possible.
For date/time, strictly follow the format "15 July 2023"...

QUESTION: {question}
SUMMARY: {summary}
Short answer:
"""

def _make_longmemeval_prompt(summary: str, question: str) -> str:
    return f"""\
You are an intelligent memory assistant tasked with retrieving accurate
information from episodic memories.

# INSTRUCTIONS:
Synthesize information from different memories to answer the user's question.
It is CRITICAL that you move beyond simple fact extraction and perform
logical inference. Answer the question in a short phrase.

QUESTION: {question}
SUMMARY: {summary}
Short answer:
"""
```

Figure 10: Python functions used to generate the final answer prompts based on context.

Write-Back Grounding Verification Prompt

You are a strict quality judge for memory facts. Given a question and candidate facts, select ONLY high-quality facts that meet ALL criteria.
Question: {question}
Candidate Facts: {facts_list}
Quality Criteria (ALL must be met):

1. **Directly Relevant**: Directly helps answer this specific question.
2. **Specific & Concrete**: Contains names, dates, numbers, locations.
3. **Factually Grounded**: Based on concrete conversation content.
4. **Non-redundant** & **Self-contained**.

Reject facts that are: Too general, obvious from the question, vague, or tangentially related.
Output format: Return ONLY a JSON array of selected fact indices (0-based), e.g., [0, 2], [1], or [] if none qualify.

Figure 11: Prompt for verifying the quality and relevance of extracted facts.

LLM as Judge Evaluation Prompt

Your task is to label an answer to a question as 'CORRECT' or 'WRONG'. You will be given: (1) a question (posed by one user to another user), (2) a 'gold' (ground truth) answer, (3) a generated answer.
The gold answer will usually be a concise and short answer. The generated answer might be much longer, but you should be generous with your grading - as long as it touches on the same topic/time as the gold answer, it should be counted as CORRECT.
Question: {question} Gold answer: {gold_answer} Generated answer: {generated_answer}
First, provide a short (one sentence) explanation of your reasoning, then finish with CORRECT or WRONG in a json format with the key as "label".

Figure 12: The LLM-as-a-Judge prompt used for automated evaluation.