Reuse, Don't Recompute: Efficient Large Reasoning Model Inference via Memory Orchestration

Daivik Patel
Independent
daivik.d.patel@gmail.com

Shrenik Patel
Independent
shrenik.d.patel@gmail.com

Abstract

Large reasoning models (LRMs) achieve strong accuracy through test-time scaling (TTS), generating longer chains of thought or sampling multiple solutions, but at steep costs in tokens and latency. We argue that memory is a core ingredient for efficient reasoning: when evidence already exists, models should "think less" by reusing structured memory instead of recomputing derivations. We present ENGRAM-R, an inference-time memory layer that integrates typed retrieval with compact fact card representations and explicit citation control. On the LoCoMo benchmark, ENGRAM-R reduces input tokens by 85% and reasoning tokens by 75% versus full context while maintaining high accuracy. On a multi-hop slice of the LongMemEval benchmark, it achieves similar efficiency with substantial accuracy gains. These results show that memory is not only critical for long-horizon correctness, but also a practical lever for efficient reasoning under tight compute, memory, and latency budgets.

1 Introduction

Large reasoning models (LRMs) have made "thinking longer" a default recipe for better answers: expand the chain-of-thought (CoT) [19], sample more drafts [18], and hope consensus emerges. Although this principle yields in real gains, it doesn't come without costs. Lengthy test-time reasoning inflates latency, burns tokens on evidence the model already "knows," and turns deployment into an exercise in budget triage rather than engineering. If LRMs are to be used in constrained settings, we need a way *to spend fewer tokens without compromising accuracy*.

We argue that the right lever is not *more* thought, but *less redundant* thought. Focusing on test-time compute, we set the goal of reducing the tokens and wall-time an LRM spends during inference. Our premise is simple: many tasks (especially those with recurring entities, facts, and routines) don't require recomputing long chains each time. Instead, they benefit from **typed, reusable evidence** that can be selectively retrieved and composed at inference. We operationalize this with a **memory-orchestrated inference** layer that sits strictly outside the model weights. It organizes interaction traces and knowledge as **episodic, semantic, and procedural** records; performs **dense, type-aware retrieval** into a small fixed budget; and ultimately helps **constrain a model's reasoning**.

Our contributions are the following. (1) We present ENGRAM-R, a **memory-orchestrated inference** recipe for LRMs that empirically reduces test-time cost without touching model weights. (2) We deliver evidence on **two long-horizon benchmarks** that typed retrieval can replace large fractions of CoT while preserving fidelity, along with a clear diagram and reasoning trace to make the process auditable.

The message is clear and actionable: *test-time scaling is not the only path to reliability*. For many real deployments, **reuse beats recompute**. Typed memory and selective retrieval let LRMs reach correctness sooner, with fewer tokens, lower latency, and tighter budgets.

2 Related Work

Research on long-term memory for language models spans non-parametric retrieval, structured graphs, and system-level abstractions. Early work coupled frozen models with dense or lexical retrieval [6, 8, 9], improving factual recall but often relying on heuristic chunking and calibration. Retrieval-pretrained and nearest-neighbor approaches scale this line further [4], trading simplicity for freshness and editability. Structured methods instead frame memory as a graph, capturing entities and relations to support multi-hop reasoning [1, 10], while system-level proposals treat memory as a schedulable resource with paging and lifecycle management [11, 16]. These designs extend capacity but typically return verbose snippets, leaving models to repeatedly re-narrate evidence.

Parallel to advances in memory, work on LRMs has begun to emphasize efficiency. While early advances in LRMs emphasized thinking longer through extended chains or ensemble-style decoding, more recent work has shifted toward making inference more efficient. Methods such as *Skeleton-of-Thought* [13] and *Chain-of-Draft* [21] restructure reasoning to reduce redundancy and accelerate generation, showing that careful control over inference can yield accuracy at lower cost. Our approach is complementary: rather than restructuring the chain itself, we compress the evidence surface, reducing the need for redundant reasoning by constraining what models are allowed to see and cite.

By re-rendering retrieved content into compact, traceable *Fact Cards*, ENGRAM-R enables LRMs to handle long-horizon reasoning with **sharply reduced token and latency budgets** while maintaining competitive accuracy—a practical path to efficient reasoning at scale.
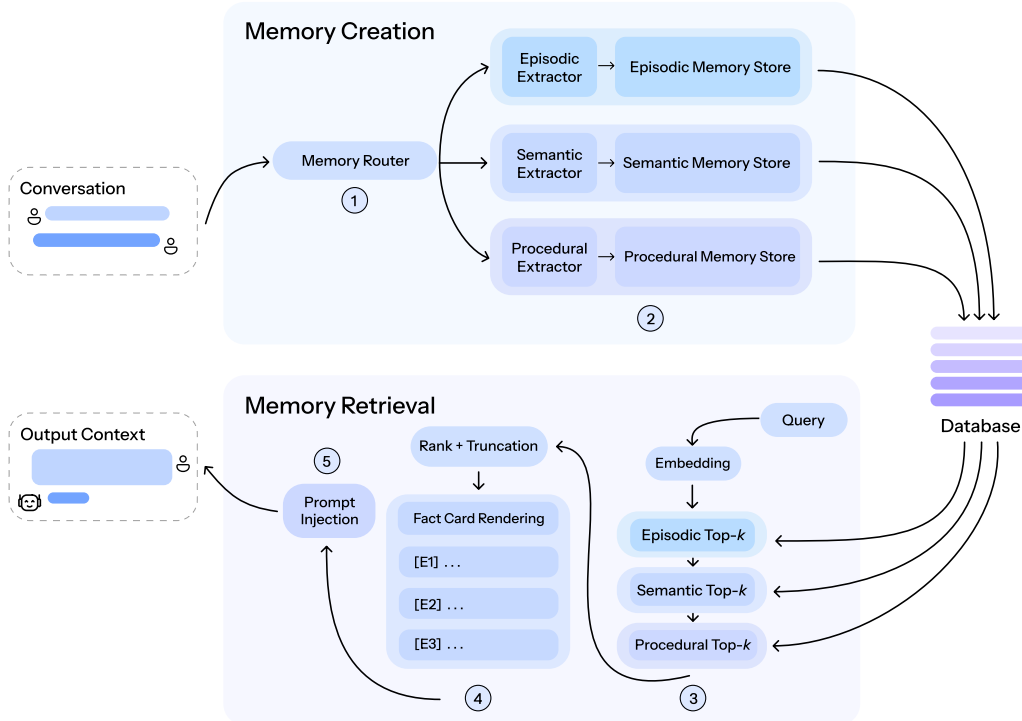
3 The ENGRAM-R Architecture

Figure 1: **System overview of ENGRAM-R.** Dialogue turns are routed into typed stores (episodic, semantic, procedural), embedded, and retrieved at query time. Retrieved items are re-rendered into compact *Fact Cards*, which provide atomic, anchored claims for direct citation. This design bounds input size, shortens reasoning, and enables LRMs to achieve efficient long-horizon inference. Numbers (1)–(5) mark the main components and are referenced below.

We extend the ENGRAM memory architecture [2] into a full **efficiency layer for LRMs**. Analogous to ENGRAM, our system organizes dialogue into typed memory stores [5, 17] and performs dense retrieval at query time, but we add two critical extensions: (i) retrieved items are re-rendered into compact *Fact Cards*, and (ii) models are instructed to *cite cards directly* in their reasoning trace. These additions turn memory from a verbose context dump into a bounded, auditable evidence substrate. The pipeline consists of five main stages, marked (1)–(5) in Figure 1 and are referenced throughout this section. Here, we briefly recap the essentials and then introduce our extensions.

At a high level, ENGRAM routes each dialogue turn into one or more typed stores (episodic, semantic, procedural) (1), normalizes it into a lightweight record paired with an embedding, and persists it in a relational memory (2). At query time, the system retrieves top-k candidates from each store using cosine similarity, merges and deduplicates them, and truncates to a fixed budget (typically $K=25$) (3). A full specification of the original ENGRAM architecture we extend is in Appendix A.

3.1 Problem Setup

We study the problem of *efficient long-horizon reasoning* in LRMs. In this setting, a dialogue unfolds turn by turn, with each entry consisting of a speaker, their dialogue, and a timestamp. At query time, the model must answer a question that may depend on information from any point in the conversation, even far back in history.

A naive baseline is to pass the *entire conversation* as input. This guarantees recall but scales poorly: tokens grow linearly with turns, often exceeding context limits. Test-time scaling (TTS) strategies such as extended chains [19] or self-consistency [18] amplify cost further, inflating both latency and token budget. To avoid this, ENGRAM uses retrieval:

$$\tilde{R}(q) \; = \; \mathrm{TopK}\{\, m \in \mathcal{M} \mid \mathrm{score}(q, m) \,\}, \tag{1}$$

where $\mathcal{M}$ is the typed memory state, score is a similarity function, and $\tilde{R}(q)$ is a compact set of candidates that ensures coverage without replaying the full dialogue.

The limitation is that even $\tilde{R}(q)$ contains multi-sentence, verbose snippets. Passing these to an LRM forces it to re-narrate evidence before using it, wasting tokens and elongating reasoning chains. Our contribution is to transform $\tilde{R}(q)$ into compact, transparent *Fact Cards*, paired with a controlled citation mechanism, enabling LRMs to consume retrieval efficiently and within bounded reasoning cost.

3.2 Extensions to ENGRAM

Our contribution extends this pipeline at two key stages:

1. Aggregation → Fact Card Rendering. Rather than injecting raw, often verbose record text, we re-render each retrieved memory $m \in \tilde{R}(q)$ into a compact **Fact Card** (4). A Fact Card captures only the essentials of a record

$$\phi(m) \; = \; \big[\, \mathrm{id}(m), \; \mathrm{claim}(m), \; \mathrm{anchor}(m) \,\big]$$

where $\mathrm{id}(m)$ is a stable identifier (e.g., [E1], [E2]), $\mathrm{claim}(m)$ is a minimal canonical statement distilled from the record, and $\mathrm{anchor}(m)$ is a timestamp or provenance marker that grounds the claim. Formally, the set of retrieved records $\tilde{R}(q)$ is transformed into a new evidence state

$$\mathcal{F}(q) \; = \; \{\phi(m) \mid m \in \tilde{R}(q)\}.$$

Intuitively, $\mathcal{F}(q)$ is a set of atomic, non-redundant claims tied to explicit provenance. This step compresses multi-sentence snippets into concise, verifiable evidence units that the model can cite directly, while preserving the information required for faithful reasoning.

2. Prompt Construction → Controlled Citation. In the final stage, the answering model is conditioned not on raw text but on the compact set of Fact Cards $\mathcal{F}(q)$ (5). To ensure accountability, we add a *controlled citation mechanism*: the model must reference evidence by explicit identifiers rather than paraphrasing. Formally, the prompt is constructed as

$$P(q) \; = \; \mathrm{Template}\big(q, \mathcal{F}(q)\big)$$

where the template specifies that answers must cite card IDs (e.g., [E1], [E2], ...) when justifying claims. At inference, the generated answer $\hat{a}$ is required to satisfy

$$\hat{a} \;\Rightarrow\; \big\{\, [E_i] \;\big|\; \phi(m_i) \in \mathcal{F}(q) \,\big\},$$

meaning that any citation in $\hat{a}$ must correspond to a valid Fact Card in the retrieved set. This constraint bounds reasoning length by preventing the model from re-describing evidence, while also making outputs *auditable*: every justification can be traced back to a specific, compact Fact Card.

Overview. Together, Fact Cards and controlled citation reframe ENGRAM from a generic memory module into an *efficiency layer for LRMs*. By compressing evidence into $\phi(m)$ and constraining its use through explicit citation, we reduce both input and reasoning cost while maintaining fidelity. Importantly, this provides a transparent efficiency layer: answers are short, accountable, and directly linked to their supporting records. For a step-by-step walkthrough of this process, see Appendix D.

4 Evaluation

We evaluate ENGRAM-R on two complementary long-horizon conversational benchmarks. LoCoMo compresses realistic two-speaker dialogues into long, multi-session conversations spanning diverse reasoning categories; LongMemEval$_S$ embeds QA in extended user–assistant histories that stress multi-session and temporal reasoning. Our evaluation protocol reports judge-based answer quality alongside input/reasoning tokens and p50/p95 latency. Numerical results are detailed in the next section.

4.1 Benchmarks

LoCoMo. LoCoMo consists of multi-session dialogues built through a human–machine pipeline grounded in personas and event graphs, then refined by human annotators for long-range consistency [12]. It contains 10 dialogues, each averaging 600 turns and $\approx$16K tokens across up to 32 sessions. The QA split covers five categories: single-hop, multi-hop, temporal, open-domain, and adversarial. In line with prior work, we exclude adversarial cases when reporting QA metrics and present category-level results for the remaining four.

LongMemEval$_S$. LongMemEval is designed to test interactive memory in user–assistant dialogues, covering five abilities: information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention (declining to answer when evidence is insufficient) [20]. It contains 500 curated questions embedded in chat histories of configurable length. For our study, we focus on LongMemEval$_S$ ($\approx$115K tokens per problem) and report QA metrics on the multi-session reasoning and temporal reasoning categories. These settings are particularly challenging and better highlight whether memory orchestration can deliver efficiency gains without sacrificing accuracy.

4.2 Backbone and baselines

All experiments are conducted with `gpt-oss-20b` [15], selected as a strong open-source LRM with competitive chain-of-thought capabilities. The model weights remain frozen; our interventions act only at inference time. We evaluate two inference settings:

1. **Full-Context baseline.** The model is provided with the complete dialogue history for each query and allows unconstrained reasoning, representing the standard LRM-only deployment approach.

2. **ENGRAM-R.** This setting augments inference with a compact memory orchestration layer: interaction traces and knowledge are organized into typed records; for a given query, we perform dense retrieval and truncate to a fixed evidence budget K before presenting this condensed context to the model. The memory layer runs with `gpt-4o-mini` [14], a non-reasoning, intentionally fast and lightweight model, as its backbone, conserving reasoning for the question-answering stage.

4.3 Metrics

To evaluate the performance of our approach, we consider two primary dimensions that capture both the quality of the output and the computational efficiency of the system:

1. **Quality.** We use an **LLM-as-Judge** protocol [22] as the main metric of semantic correctness. This involves using an independent LLM that, given the question, gold answer, and prediction, renders a binary semantic-correctness decision based on factual fidelity, completeness, and contextual appropriateness of the response. This ensures robustness to surface-level variation in responses.

2. **Efficiency.** We decompose efficiency into:

   - **Input tokens:** the number of tokens supplied to the model (prompt + retrieved evidence), reflecting context compression.
   - **Reasoning tokens:** the number of tokens generated as chain-of-thought, reflecting the cost of inference-time reasoning.
   - **Latency:** reported as end-to-end wall-clock time per query with p50 and p95 to capture both typical and tail behavior.

These metrics provide a comprehensive assessment of both the model's reasoning capabilities and its operational efficiency. By examining quality alongside efficiency, we gain a deeper understanding of how well ENGRAM-R balances performance with computational cost, offering valuable insights for real-world deployment scenarios.

5 Results

In this section, we present a comprehensive evaluation of ENGRAM-R across multiple datasets and reasoning tasks. The results highlight how the proposed approach consistently improves both efficiency and accuracy, particularly in scenarios requiring long-horizon reasoning, without sacrificing performance.

Table 1: **LoCoMo results**: input tokens, reasoning tokens, and judge accuracy across QA categories. ENGRAM-R yields large efficiency gains while improving multi-hop and temporal accuracy.

Category	Setting	Input tokens	Reasoning tokens	LLM judge (%)
Single-hop	Full-Context	15,614,211	698,845	84.7
	ENGRAM-R	1,802,531	199,718	79.1
Multi-hop	Full-Context	5,187,624	300,631	72.0
	ENGRAM-R	602,634	81,035	74.5
Temporal	Full-Context	5,786,564	271,193	67.3
	ENGRAM-R	686,947	74,337	69.2
Open-domain	Full-Context	1,783,304	65,319	64.6
	ENGRAM-R	201,366	23,334	57.2
Overall	Full-Context	28,371,703	1,335,988	77.5
	ENGRAM-R	**3,293,478**	**378,424**	**75.6**

5.1 LoCoMo: Category-level Analysis

Table 1 reports LoCoMo results by QA category. The data conveys how ENGRAM-R consistently **reduces inputs by ≈89%** across categories and **shrinks reasoning token usage by ≈72%**. Crucially, accuracy is *maintained or improved* precisely where long-range composition matters most: **multi-hop (+2.5%)** and **temporal (+1.9%)**. These gains support the hypothesis that typed memory plus compact, citable evidence reduces the model's need to re-derive multi-step chains already present in the dialogue history. It is also evident that two categories show accuracy headroom. Single-hop (short, localized queries) experiences a slight decrease, which aligns with the idea that heavy truncation may not be necessary for questions focused on immediate context. The largest difference appears in open-domain settings, where having access to broader background knowledge would potentially be useful.

Table 2: **LongMemEval$_S$ results**: compact typed evidence improves efficiency and accuracy on very long histories.

Category	Setting	Input tokens	Reasoning tokens	LLM judge (%)
Multi-session	Full-Context	13,741,996	94,915	36.8
	ENGRAM-R	631,083	20,920	66.9
Temporal	Full-Context	13,741,916	154,180	39.1
	ENGRAM-R	602,173	33,831	52.6
Overall	Full-Context	27,483,912	245,125	38.0
	ENGRAM-R	**1,233,256**	**54,301**	**59.8**

5.2 LongMemEval$_S$: Performance Analysis

Table 2 summarizes performance on the LongMemEval$_S$ slice that includes the multi-session and temporal reasoning categories. Full-context accuracy is low despite unbounded inputs, suggesting that extremely long histories hinder effective use of context in LRMs. ENGRAM-R reverses this: by condensing and typing prior information into compact, auditable evidence, the model avoids being lost in a sea of context (100k+ tokens) and instead is able to cite what matters. The result is a **+21.8% overall accuracy** improvement alongside $\approx$**96% fewer input tokens** and $\approx$**78% fewer reasoning tokens**. As mentioned, the gains are incredibly evident in both the **multi-session (+30.1)** and **temporal reasoning (+13.5)** categories, where durable memory and temporal anchors are essential for efficient reasoning.

Table 3: **Search and total inference latency (seconds).** We report median (p50) and tail (p95) latencies across datasets. Dashes indicate that search latency is not applicable in the full-context baseline.

Dataset	Setting	Search		Total	
		p50	p95	p50	p95
LoCoMo	Full-Context	–	–	7.89	17.16
	ENGRAM-R	**1.04**	**2.67**	**2.56**	**7.43**
LongMemEval$_S$	Full-Context	–	–	9.62	21.47
	ENGRAM-R	**0.72**	**1.18**	**1.88**	**5.54**

5.3 Latency and Tail Behavior

Latency trends in both methods follow token reductions (Table 3). Relative to full-context, ENGRAM-R **reduces median total latency** during QA time by $\approx$**68%** on LoCoMo and $\approx$**81%** on LongMemEval$_S$. Tail latency (p95) sees improvements on a similar magnitude. The memory search stage itself is fast, with p50 times of ≤ 1.04 s, indicating that the retrieval process does not introduce much overhead. The primary efficiency gains, however, come from the reduction in both the input context size and the reasoning tokens generated during the answering process. By limiting the model's input and focusing on the most relevant information, ENGRAM-R effectively shrinks the reasoning footprint, leading to faster response times.

5.4 Summary and Implications

ENGRAM-R consistently delivers **order-of-magnitude input compression and large reductions in reasoning tokens**. Across datasets and metrics, the findings support the effectiveness of compact, typed memory as a means to **reduce test-time compute** in LRMs without compromising fidelity on tasks requiring long-horizon reasoning. ENGRAM-R maintains strong accuracy on LoCoMo while materially reducing both the context ingested and the reasoning effort required. On LongMemEval$_S$, where dialogue histories are especially long, typed memory transforms unwieldy transcripts into traceable, citable evidence, reversing the typical failure mode of full-context inference. These reduc-

tions in input and generated tokens also translate directly to wall-clock gains, including significant drops in both median and tail latency during the answering stage.

5.5 Practical application

Beyond long-context stress tests, we evaluate a deployment-style scenario in healthcare, using the *HealthBench* benchmark [3], to assess whether inference-time efficiency transfers to a high-stakes domain with *shorter*, multi-turn inputs. Even though the room to compress *reasoning* is smaller, ENGRAM-R consistently trims generative overhead while keeping accuracy near parity, with gains on temporally anchored questions. Full results and a separate discussion appear in **Appendix C**.

6 Discussion and Conclusions

ENGRAM-R reframes external memory not merely as a way to increase recall, but as a **mechanism for spending less compute to reach the same (or higher) fidelity**. In our setting, the practical consequence is visible in both token budgets and wall-clock time: the model reads far less, writes far less, and still answers competitively on categories/questions that require long-horizon composition with reasoning capabilities.

Benefits of typed representation. Two design decisions appear central. First, *typing* (episodic, semantic, procedural) reduces competition among heterogeneous content and produces evidence sets that are semantically coherent for the downstream reasoner; this is consistent with the gains on multi-hop and temporal categories, where composition depends on keeping timelines and stable facts disentangled. Second, *compact, anchored rendering* (Fact Cards) creates a low-friction interface between retrieval and inference: the model does not need to re-narrate evidence it already "has," and the citations are reviewable by construction.

Limitations. There are several. (i) *Coverage vs. compactness.* An aggressive evidence budget can miss rare but decisive facts; conservative budgets blunt efficiency. Budget selection remains task-dependent; thus it may be useful to have a dynamic budget. (ii) *Staleness and drift.* External memory can become outdated; without refresh policies or validity intervals, models may confidently cite obsolete cards. (iii) *Evaluation bias.* LLM-as-judge captures semantic agreement but is not perfect; a potential mitigation may include blinded human evaluation on LLM-graded samples.

Future work. The results suggest several immediate directions for the efficient reasoning community: (1) *Adaptive budgets and routing.* Learn a small controller that sets per-query (or per-type) budgets under a global compute cap, using features from retrieval confidence, novelty, and recency. (2) *Temporal consistency checking.* Equip the memory layer with lightweight temporal constraint solvers that sanity-check card sets before answering (preventing contradictions that would otherwise require long generative reconciliation). (3) *Cross-modal and multilingual memory.* Extend cards to admit visual or tabular anchors and evaluate whether typed reuse equally reduces compute for multimodal LRMs.

Conclusions. Reducing test-time compute lowers the barrier to deploying reasoning-capable systems in resource-constrained environments such as edge devices, NGOs, and educational settings. In long-horizon tasks, ENGRAM-R exemplifies how structuring prior interactions into compact, typed, and citable evidence enables LRMs to **answer with fewer tokens and reduced latency**, all while maintaining high fidelity on tasks requiring complex composition. This approach offers a unique perspective on efficient reasoning, emphasizing **reuse over recompute** and demonstrating that significant computational savings can be achieved without sacrificing performance. By offering a model-agnostic, auditable mechanism that lives outside model weights, we hope to encourage broader adoption of advanced LRMs and systems involving LRMs in scenarios where computational resources are limited.

References

[1] Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Mikhail Burtsev, and Evgeny Burnaev. Arigraph: Learning knowledge graph world models with episodic memory for llm agents. *arXiv preprint arXiv:2407.04363*, 2024. doi: 10.48550/arXiv.2407.04363. URL `https://arxiv.org/abs/2407.04363`.

[2] Anonymous. Engram: Effective, lightweight memory orchestration for conversational agents. Under review at International Conference on Learning Representations (ICLR), 2026.

[3] Rahul K. Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, Johannes Heidecke, and Karan Singhal. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025. URL `https://arxiv.org/abs/2505.08775`.

[4] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Ethan Rutherford, Katie Millican, Danny Driess, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022. URL `https://arxiv.org/abs/2112.04426`.

[5] Neal J. Cohen and Larry R. Squire. Preserved learning and retention of pattern-analyzing skill in amnesia: dissociation of knowing how and knowing that. *Science*, 210(4466):207–210, 1980. doi: 10.1126/science.7414331.

[6] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 3929–3938, 2020. URL `https://arxiv.org/abs/2002.08909`.

[7] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL `https://arxiv.org/abs/2004.04906`.

[8] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, and Luke Zettlemoyer. Generalization through memorization: Nearest neighbor language models. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. URL `https://proceedings.mlr.press/v119/khandelwal20a.html`.

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020. URL `https://arxiv.org/abs/2005.11401`.

[10] Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, Wenbo Su, and Bo Zheng. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12758–12786, 2024. doi: 10.18653/v1/2024.findings-emnlp.746. URL `https://aclanthology.org/2024.findings-emnlp.746`.

[11] Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, et al. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*, 2025. doi: 10.48550/arXiv.2507.03724. URL `https://arXiv.org/abs/2507.03724`.

[12] 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 (ACL), Long Papers*, pages 13851–13870, 2024. URL `https://aclanthology.org/2024.acl-long.747.pdf`.

[13] Xuezhi Ning, Ye Liu, Ning Ding, Zhiheng Wang, Yingheng Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. Skeleton-of-thought: Large language models can do parallel decoding. *arXiv preprint arXiv:2307.15337*, 2023. URL `https://arxiv.org/abs/2307.15337`.

[14] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. doi: 10.48550/arXiv. 2303.08774.

[15] OpenAI. Gpt-oss-120b & gpt-oss-20b: Openai's open-weight reasoning models. 2025.

[16] Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2307.05030*, 2023. doi: 10.48550/arXiv.2307.05030. URL `https://arxiv.org/abs/2307.05030`.

[17] Endel Tulving. Episodic and semantic memory. In Endel Tulving and Wayne Donaldson, editors, *Organization of Memory*, pages 381–403. Academic Press, 1972.

[18] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL `https://arxiv.org/abs/2203.11171`.

[19] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24824–24837, 2022. URL `https://arxiv.org/abs/2201.11903`.

[20] Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Long-memeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024. URL `https://arxiv.org/abs/2410.10813`.

[21] Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less. *arXiv preprint arXiv:2502.18600*, 2025. URL `https://arxiv.org/abs/2502.18600`.

[22] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023. URL `https://arxiv.org/abs/2306.05685`.

A ENGRAM Base Architecture

This appendix summarizes the *base* ENGRAM system used throughout the paper (i.e., *without* the fact card and citation controls introduced in Section 3). The goal is to present a compact, reproducible description of the original pipeline and data structures that were used for evaluating memory in non-reasoning models. The numbers in the subheadings correspond to labeled steps in Figure 1.

A.1 System overview

ENGRAM converts a multi-turn dialogue into a durable, typed memory and, at query time, retrieves a small set of relevant records to condition an answering model. The pipeline comprises five components:

(1) **Routing.** Decide which memory types an incoming turn should update.

(2) **Extraction & embedding.** Normalize selected content into a record schema and compute an embedding.

(3) **Typed retrieval.** At query time, retrieve top-k candidates *within each* memory type by dense similarity.

(4) **Aggregation.** Merge per-type candidates, deduplicate, and truncate to a fixed budget K.

(5) **Prompt construction.** Serialize the selected records and inject them as context to the answering LLM.

A.2 Dialogue and memory notation

Let a dialogue be $\mathcal{C} = \{x_t\}_{t=1}^T$, with $x_t = (s_t, u_t, \tau_t)$ denoting speaker identity $s_t \in \{A, B\}$, utterance text u_t, and timestamp $\tau_t \in \mathbb{R}_+$. ENGRAM maintains a typed memory state

$$\mathcal{M} = (\mathcal{M}_{\text{epi}}, \mathcal{M}_{\text{sem}}, \mathcal{M}_{\text{pro}}),$$

to support answering queries q posed after the dialogue unfolds.

A.3 Routing and storage (1–2)

A lightweight router maps each utterance to a compact three-bit decision

$$r(u_t) \in \{0, 1\}^3 \implies b_t = (b_t^{\text{epi}}, b_t^{\text{sem}}, b_t^{\text{pro}}),$$

indicating whether to update the *episodic*, *semantic*, and/or *procedural* stores. For each type flagged by b_t, ENGRAM constructs a normalized record and pairs it with an embedding $e \in \mathbb{R}^d$ from an encoder $g : \mathcal{U} \to \mathbb{R}^d$. Records and vectors are persisted in a simple relational store (SQLite), keyed by conversation and type. The router's 3-bit output keeps decisions interpretable and facilitates ablations.

A.4 Typed record schemas (2)

ENGRAM uses type-specific, minimally sufficient schemas:

$$
\begin{aligned}
m^{\text{epi}} &= (t, \ \sigma, \ \delta, \ e) && \text{(event title, brief summary, time anchor, embedding)} \\
m^{\text{sem}} &= (f, \ \delta, \ e) && \text{(fact string, time anchor, embedding)} \\
m^{\text{pro}} &= (t, \ c, \ \delta, \ e) && \text{(procedure title, normalized content, time anchor, embedding)}
\end{aligned}
$$

Typed separation reduces competition during retrieval (events vs. stable facts vs. procedures) and exposes structure that is easy to audit.

A.5 Dense retrieval and budgeted aggregation (3–4)

Given a query q, ENGRAM embeds it as $e_q = g(q)$ and retrieves candidates *within each* store by cosine similarity [7]

$$R_k(q) = \text{TopK}\{\text{score}(e_q, m) \mid m \in \mathcal{M}_k\}, \quad k \in \{\text{epi}, \text{sem}, \text{pro}\}.$$

Per-type sets are then merged and deduplicated across stores, and the union is truncated to a fixed evidence budget K

$$\tilde{R}(q) = \text{Truncate}_K \Big(\text{Dedup} \big(\bigcup_k R_k(q) \big) \Big).$$

In all experiments we default to $K=25$, a knee point identified by a K-sweep ablation performed in the ENGRAM paper [2].

A.6   Prompt construction and answering (5)

For multi-speaker dialogues, ENGRAM materializes speaker-specific banks $\tilde{R}(q, A)$ and $\tilde{R}(q, B)$. Each record m is serialized with its temporal anchor

$$\ell(m) = \delta(m) \ : \ \text{text}(m).$$

A deterministic formatting function $\text{Template}(\cdot)$ combines the query and serialized records into the final prompt

$$P(q) = \text{Template}\Big(q, \{\ell(m)\}_{m \in \tilde{R}(q,A)}, \{\ell(m)\}_{m \in \tilde{R}(q,B)} \Big),$$

which is then passed to the answering model to obtain $\hat{a} = \text{LLM}\big(P(q)\big)$. Separating banks by speaker helps preserve attribution, which is especially important during answering time.

A.7   Remarks and scope

This appendix describes *base ENGRAM*: routing, typed storage, dense per-type retrieval, budgeted aggregation, and prompt injection of *serialized snippets*. Section 3 of the main paper extends the *representation and control* of the retrieved set (re-rendering into compact fact cards and enforcing citation), while leaving routing, storage, and retrieval unchanged.

A.8   Reproducibility

To ensure transparency and reproducibility of our results, we provide the complete codebase used to implement the ENGRAM-R system, as well as all scripts required to reproduce the experiments on the LoCoMo and LongMemEval benchmarks.

The code and documentation are publicly available at the following anonymous repository: `https://anonymous.4open.science/r/engram-r-7F5A/README.md`

B ENGRAM-R Architecture Ablation

Experiment Setup. This ablation isolates the contribution of the fact cards and citation enforcement layer introduced in this paper. Thus we compare **ENGRAM Base** against **ENGRAM-R** and hold all components but the architecture fixed: same backbone LRM, same router, same typed stores, same dense retriever, and the same evidence budget ($K = 25$). Experiments are run on LoCoMo with the QA categories used in the main results.

Table 4: LoCoMo Results: ENGRAM Base

Category	Input tokens	Reasoning tokens	Judge (%)
single-hop	1,743,554	380,007	78.3
multi-hop	597,306	170,232	66.6
temporal reasoning	571,784	130,740	61.5
open-domain	199,853	42,998	71.5
overall	3,112,497	723,997	73.5

Table 5: LoCoMo Results: ENGRAM-R

Category	Input tokens	Reasoning tokens	Judge (%)
single-hop	1,802,531	199,718	79.1
multi-hop	602,634	81,035	74.5
temporal reasoning	686,947	74,337	69.2
open-domain	201,366	23,334	54.2
overall	3,293,478	378,424	74.6

Findings. Tables 4, 5 show a clear pattern. Moving from ENGRAM Base to ENGRAM-R **nearly halves reasoning tokens** overall (-47.8%), with small changes in input size and a modest improvement in judge accuracy (+1.1%). The largest quality gains appear exactly where long-horizon composition is required: **multi-hop (+7.9%)** and **temporal reasoning (+7.7%)**, alongside sharp reductions in reasoning effort. For single-hop questions, accuracy is similar while reasoning is substantially shorter. The open-domain slice is the notable exception: while reasoning tokens shrink by $\approx 46\%$, accuracy drops (–17.3%), consistent with the need for broad background that is not captured in typed stores or that benefits from wider context.

Interpretation. ENGRAM-R trades a *small, predictable* increase in input tokens (card headers/anchors and citation scaffolding) for a **large, robust decrease in generated reasoning**. The explicit instruction to *cite* cards by anchor eliminates re-narration of retrieved content, and the atomic rendering reduces the need for the model to reconstruct intermediate steps already present in memory. In categories that rely on stitching together dispersed events or facts over time, this *representation + control* acts like a budgeted controller: the model can answer succinctly by pointing to the right cards.

Takeaways. The ablation justifies the new components introduced in this paper: (i) *Fact Cards* convert retrieved snippets into compact, auditable units that are easy for the model to reuse; and (ii) *Citation Enforcement* turns those units into a control surface that shortens chains without degrading fidelity on long-horizon reasoning. Under identical retrieval budgets and decoding, the combination achieves the intended goal of **reducing test-time compute at inference** while maintaining (and often improving) answer quality.

C HealthBench Evaluation

Motivation and setup. *HealthBench* targets challenging, clinically relevant question answering and reasoning. We include it to test whether *inference-time* efficiency mechanisms carry over to an impactful domain with domain-specific structure. We compare **full-context** against **ENGRAM-R** under the same backbone LRM, identical decoding, and the same retrieval budget K as in the main experiments. Evaluation uses the same LLM-as-Judge protocol and token accounting (input vs. reasoning) as in Section 5.

We evaluate on a **subset of HealthBench**, restricted to categories with *multi-turn inputs*, where memory and control mechanisms are most relevant. This subset includes *context-seeking*, *complex-responses*, and *health-data*.

Why HealthBench differs from LoCoMo/LongMemEval. Unlike *LoCoMo* and *LongMemEval*, HealthBench is *not* designed as a long-context memory stress test. There are multi-turn inputs, but histories are *much shorter* than the 10k–100k token conversations seen in our main benchmarks. Two consequences follow: (i) the fixed overhead of rendering *fact cards* (anchors/headers) can slightly *increase* input tokens when the original context is already compact, and (ii) the achievable reduction in *reasoning* tokens is less pronounced, because there is less redundant narration to remove. We therefore expect smaller efficiency deltas than on LoCoMo/LongMemEval, with accuracy near parity if typed evidence remains sufficient.

Table 6: **HealthBench results:** input tokens, reasoning tokens, and judge accuracy. ENGRAM-R rows are shaded light gray.

Category	Setting	Input tokens	Reasoning tokens	LLM judge (%)
context-seeking	Full-Context	111,429	413,653	37.4
	ENGRAM-R	121,085	271,356	35.4
complex-responses	Full-Context	120,964	310,730	32.9
	ENGRAM-R	133,423	246,423	32.4
health-data	Full-Context	95,817	327,982	37.2
	ENGRAM-R	100,392	282,136	39.5
overall	Full-Context	328,210	1,052,365	35.8
	ENGRAM-R	354,900	799,915	35.7

Results and interpretation. ENGRAM-R **reduces reasoning tokens overall by roughly ≈24%**. *Input tokens* slightly increase (+8% overall), and accuracy remains at parity overall ($35.8\% \rightarrow 35.7\%$), with a notable improvement on **health-data** (+2.3%), which benefits from explicit temporal anchors in typed memory. The mild drops on **context-seeking** and **complex-responses** suggest that, in compact settings, broader surface context can occasionally help; a hybrid policy that permits a small untyped backfill would likely recover this gap without changing the compute profile materially.

Takeaways for efficient reasoning. Even when long-context savings are structurally limited, *representation and control* still reduce generative overhead: citation discourages re-narration, shrinking reasoning while keeping quality steady. In healthcare-adjacent tasks, this matters for *latency* and *throughput* under budget constraints.

D Full Reasoning–QA Walkthrough

Worked Example.

To illustrate the ENGRAM-R pipeline, we present a compact reasoning walkthrough.

Dialogue. The conversation unfolds as follows:

Turn	Speaker	Utterance
1	A	"After months of searching for a new role and packing up my old apartment, I finally *moved to Seattle* last year. It took a while to adjust, but I'm starting to feel at home in the city."
2	B	"That's exciting. Just don't forget to *file your tax return before April 15*—the deadline is strict and missing it could cause penalties."
3	A	"Appreciate the reminder. I've been decorating my new place, and I realized my *favorite color is green*; it shows up in most of the furniture and curtains."

At query time, the model receives the question:

$$q = \text{"Where does A live?"}$$

Pipeline Steps.

1. **Retrieval.** Relevant records are selected from episodic (relocation event), procedural (tax deadline), and semantic (favorite color) stores.
2. **Aggregation.** The retrieved records are merged into $\tilde{R}(q)$.
3. **Fact Card Rendering.** Each record $m \in \tilde{R}(q)$ is re-rendered into a compact representation:

$$\phi(m_1) = [E1, \text{ "A moved to Seattle"}, \ \tau{=}2024]$$
$$\phi(m_2) = [E2, \text{ "B reminded A to submit tax form by Apr 15"}, \ \tau{=}2024]$$
$$\phi(m_3) = [E3, \text{ "A's favorite color is green"}, \ \tau{=}2024]$$

4. **Prompt Construction.** The set $\mathcal{F}(q) = \{\phi(m)\}$ is inserted into a template with citation instructions.
5. **Answering.** The model generates an evidence-cited answer.

Reasoning Trace.

> *Need to answer Q1. E1 shows A relocated to Seattle. Answer: A lives in Seattle. Cite [E1].*

Model Output.

$$\hat{a} = \text{"A lives in Seattle [E1]."}$$

Takeaway. This walkthrough shows how verbose dialogue turns are condensed into *atomic, citable Fact Cards*. Instead of re-narrating evidence, the model directly cites compact evidence, yielding short and accountable reasoning traces.

Reasoning Prompt

You are an intelligent memory assistant with access to conversation memories and citation facts.
MEMORY CONTEXT (rich information)

CITATION FACTS (for referencing)

INSTRUCTIONS

1. Use the rich memory context above to understand the full situation.

2. Answer concisely in 1–3 sentences based on the memory context.

3. Cite supporting citation facts using `[E1]`, `[E2]` format.

4. In your reasoning chain, use minimal tokens by:
   - ALWAYS reference the question as "Q1" (never repeat the full question).
   - ALWAYS cite facts by label only (e.g., "E1 shows...", "E2 indicates...").
   - NEVER repeat full fact content in reasoning.
   - Be extremely concise and focused.

5. Only say "not enough info" if truly no relevant information exists.

REASONING EXAMPLE

Need to answer Q1. E1 shows Melanie ran charity race on May 20. E2 indicates Caroline was proud. Answer: May 20, 2023. Cite [E1].

QUESTION TO ANSWER

Q1: Where does A live?

Please provide your answer with reasoning