Exploring Cross-Scenario Generality of Agentic Memory Systems: Diagnostics and a Strong Baseline*

Zhikai Chen[1†] Jialiang Gu[2†] Junyu Yin[2†] Xianxuan Long[1] Shenglai Zeng[1]
Xiaoze Liu[3] Kai Guo[1] Keren Zhou[2] Jiliang Tang[1]

[1]Michigan State University [2]George Mason University [3]Purdue University

Abstract

LLM agents accumulate interaction histories that exceed their context windows, motivating extensive research on memory systems. Yet most existing designs are tailored to a single scenario (such as multi-session chat or a specific trajectory format), offering little evidence that they generalize across the heterogeneous trajectories encountered in deployment. We evaluate twelve baselines spanning specialized memory systems, long-context models, and an unindexed search agent across five scenarios: single-turn QA, multi-session chat, agentic-trajectory QA, memory stress tests, and long-horizon agentic tasks. Notably, DCI-Lite, an unindexed baseline that searches raw text via grep and bounded reads, is among the strongest existing baselines. This suggests that memory performance depends partly on granting the agent active control over retrieval rather than coupling a static store to a single retrieval route. We instantiate this diagnostic insight in **AutoMEM**, a strong baseline that exposes both raw-text and indexed memory views through an agent-controlled interface. Across the evaluated systems, AutoMEM achieves the best mean cross-scenario rank.

1 Introduction

Large language models (LLMs) have evolved from single-turn chatbots (Ouyang et al., 2022) into general-purpose agents that plan, invoke tools, and execute long-horizon workflows (Yao et al., 2023; Schick et al., 2023; Wang et al., 2024). Across these settings, models accumulate historical information from diverse sources: knowledge corpora to ground responses (Lewis et al., 2020), codebases to navigate (Jimenez et al., 2024; Heule et al., 2025), multi-step workflows to plan (Xie et al., 2024), and user state to maintain (Salemi et al., 2024). **Memory**, the mechanism that selects and organizes past information, allows an LLM to act beyond the bounds of its immediate prompt.

Many memory systems have been proposed (Hu et al., 2025), yet whether this added architectural complexity translates to robust empirical gains remains contested (Pollertlam and Kornsuwannawit, 2026; He et al., 2026). The central problem is that **most existing memory systems are scenario-narrow**: each is tailored to a single regime, typically multi-session chat (Wu et al., 2025; Maharana et al., 2024) or a specific agentic-trajectory format, and the characteristics those regimes reward diverge sharply (Zhao et al., 2026). Strong performance on one benchmark does not imply generalization to others, and little empirical evidence shows which architectural designs generalize across workloads.

We therefore revisit existing designs through the lens of **cross-scenario generality**: a practical memory system must operate effectively across the heterogeneous trajectories agents encounter in deployment, such as multi-session chat, code interactions, and browser logs. We also track token overhead and latency to ensure generality is not achieved at impractical operational cost (Yang et al., 2026).

We evaluate twelve representative baselines, including the unindexed search agent **DCI-Lite** (Li et al., 2026c), across five task families: single-turn QA, multi-session QA, agentic-trajectory QA, memory stress tests, and long-horizon agentic tasks. These baselines span long-context models, note-based stores, hierarchical multi-store architectures, graph-based systems, and RL-trained designs. Our quantitative comparison shows that DCI-Lite, which searches raw text using grep and bounded reads, is among the strongest existing baselines, whereas structured memory systems help only when their schema and retrieval interface closely align with the workload. We translate these diagnostics into the design of **AutoMEM**, which allows the agent to dynamically select between

* Code: github.com/CurryTang/memory-camera-ready
† Equal contribution.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 4722–4737
November 6-10, 2026 ©2026 Association for Computational Linguistics

raw-text and indexed views, achieving the best cross-scenario generality in our evaluation. Our key contributions are threefold:

(1) A systematic cross-scenario evaluation of twelve baselines across five task families, benchmarking task accuracy alongside token cost and inference latency.

(2) Empirical findings on agentic failure modes: On the audited agentic trajectories, we identify two measurable failure modes: a **representation-level** failure, in which build-time schemas omit step- and action-level evidence, and a **retrieval-level** failure, in which single-shot interfaces do not surface evidence retained in storage. We also find that an unindexed search agent (DCI-Lite) that defers retrieval decisions to query time is among the strongest existing baselines.

(3) **AutoMEM**, a diagnostic-driven strong baseline that improves over DCI-Lite by 22.5 points on LoCoMo and 40.0 points on LongMemEval-S, reducing overall mean rank from 5.20 to 2.75.

2 Related Work

2.1 Memory Design

Memory designs for LLM agents pursue two complementary goals: improving **quality** (accuracy, capability, generality) and improving **efficiency** (token cost, build cost, latency).

Quality. Memory design reduces to two coupled questions: how to store and update past information, and how to retrieve it. On the storage side, the central choice is granularity. The baseline stores raw chunked passages and retrieves them by similarity search (Lewis et al., 2020; Karpukhin et al., 2020), on top of which three richer designs target different access patterns: atomic notes (Chhikara et al., 2025; Zhong et al., 2024; Xu et al., 2025) make each fact an addressable, editable record, fitting evolving stable-fact workloads; OS-style tiered memory (Packer et al., 2023; Qian et al., 2025; Liu et al., 2026b) pages information through tool calls between bounded working and long-term tiers, fitting multi-session agents; and graph-based storage (Edge et al., 2024; Gutiérrez et al., 2025) makes entity-relation edges first-class, fitting multi-hop composition. On agentic tasks, however, single-granularity stores prove insufficient, and hierarchical or mixed-granularity stores (Yang et al., 2026; Yue et al., 2026; Shu et al., 2026; Zhao et al., 2026) are adopted at the cost of higher build overhead and token cost. Retrieval typically mirrors the storage form: graph traversal for graphs, semantic search for notes and passages, and tool-call paging for tiers. Beyond this default, two directions stand out: training an LLM tool-policy with RL to manage retrieval (Yue et al., 2026; Zhang et al., 2026b; Yan et al., 2025; Zhou et al., 2026; Wang et al., 2025; Yu et al., 2026) shifts memory work to post-training time at the cost of training compute, while multi-stage retrieval, the retrieval-side analog of mixed-granularity storage, ensembles multiple retrieval paths (e.g., graph traversal plus semantic search) and aggregates their candidates to recover what any single path misses, at the cost of more retrieval calls per query (Yang et al., 2026; Zhao et al., 2026).

Agent-based retrieval. Beyond static index lookup, tool-using agents reframe memory access as an active decision loop (Yao et al., 2023; Schick et al., 2023; Li et al., 2026c). Rather than relying on **passive single-shot retrieval**, the agent conducts **iterative query refinement** by inspecting intermediate evidence and dynamically issuing follow-up searches. We formalize **active retrieval control** as a key memory-system design dimension, examining how an autonomous controller evaluates intermediate evidence and arbitrates across **heterogeneous memory views**.

Efficiency. Cost-focused work groups into three patterns. *Compression* reduces content via gating, coreference resolution, and summarization (Liu et al., 2026a; Xu et al., 2026; Li et al., 2026a). *Offline delegation* shifts work to build time so retrieval becomes non-LLM (Tian et al., 2026; Gutiérrez et al., 2025; Yang et al., 2026). *Cost-tier routing* picks the cheapest module meeting a quality threshold (Zhang et al., 2026a). All three are validated mainly on conversational QA, so transfer to agentic workloads is untested. We cover the first and leave the others as future work.

2.2 Memory Benchmark

Memory benchmarks fall into three categories. **QA benchmarks** test memory through QA over conversation history or long documents (Maharana et al., 2024; Li et al., 2026b; Wu et al., 2025; Lee et al., 2025; Hu et al., 2026b; Ai et al., 2025; Pang et al., 2022; Bai et al., 2025; Hsieh et al., 2024). **Agentic QA benchmarks** ask questions about an

Scenario	Benchmark	Explanation
Personal chat assistant	LoCoMo (Maharana et al., 2024), LongMemEval-S (Wu et al., 2025)	QA over multi-session semantic/episodic memory
Large-corpus retrieval	HotpotQA (Yang et al., 2018) w/ PlugMem setup (Yang et al., 2026)	QA over a corpus beyond context size
Trajectory recall	AMABench-{ALF, Web, T2SQL} (Zhao et al., 2026)	QA over agent trajectories
Memory stress tests	MemoryAgentBench-{AR, TTL, LRU, CR} (Hu et al., 2026b)	Accurate retrieval, test-time learning, long-range understanding, conflict resolution
Real agentic tasks	ALFWorld (Shridhar et al., 2021), MemoryArena-shop / -travel (He et al., 2026)	Real-world agentic tasks

Table 1: The five task scenarios.

Category	Method(s)	Storage primitive
No memory	Long context	Full prompt
Note-based	SimpleMem (Liu et al., 2026a), Mem0 (Chhikara et al., 2025)	Atomic notes
Multi-store	PlugMem (Yang et al., 2026), LightMem (Fang et al., 2025)	Episodic + semantic + procedural stores
OS-style tiered	MemoryOS (Kang et al., 2025)	Short-, mid-, and long-term stores
Graph-based	HippoRAGv2 (Gutiérrez et al., 2025), AMA-Agent (Zhao et al., 2026)	Entity-fact KG / turn-indexed graph
Raw-text search	DCI-Lite (Li et al., 2026c), DCI-Lite+Sum (Li et al., 2026c)	Raw trajectory + grep/read agent
RL-trained	Mem-T (Yue et al., 2026), MemRL (Zhang et al., 2026b)	Trained retrieval / write policy

Table 2: Memory baselines grouped by architectural category.

agent's trajectory: AMA-Bench (Zhao et al., 2026) uses machine-generated trajectories with causal dependencies; SAGE (Hu et al., 2026a) targets retrieval in deep research agents. **Real agentic task benchmarks** couple memory to actions so memory quality affects task completion; MemoryArena (He et al., 2026) runs memory-agent-environment loops. Existing benchmarks typically target a single category and rarely report token cost or latency, leaving no principled comparison along the cross-task, cost, and latency axes.

3 Evaluation Design

We measure each memory system on three axes: cross-scenario generality, token cost, and latency.

Tasks. Table 1 lists the five scenarios, which span the main deployment regimes for memory in agentic systems.

We treat memory broadly as storage plus retrieval/control, rather than restricting it to personal-chat memory. HotpotQA isolates corpus memory over a store larger than the context window; ALFWorld tests whether procedural memories from prior trajectories help an acting policy.

Memory baselines. We evaluate nine memory systems, a long-context reference, and two configurations of an unindexed search agent (DCI-Lite) (Li et al., 2026c) originally designed for direct-corpus search (Table 2), covering the architectural design space of §2.1. For agentic tasks we use a ReAct agent (Yao et al., 2023) with task-tuned prompts.

Metrics. For LoCoMo, LongMemEval-S, HotpotQA, AMABench, and MemoryAgentBench we report a Qwen3-32B LLM-judge score (Zhao et al., 2026), since token F1 mis-scores long-form answers on verbose benchmarks (Appendix B.1). For agentic tasks we report environment outcomes: success rate on ALFWorld; process score on MA-Shop (fraction of subtasks solved); and c-sPS on MA-Travel, a corrected MemoryArena sPS (He et al., 2026) that only scores slots the user asks about in the current session. Appendix C evaluates judge and human agreement.

Backbones. The default backbone is Qwen3-32B (Yang et al., 2025) for both construction and answering, with Qwen2.5-7B-Instruct on ALFWorld (Qwen3-32B solves most episodes trivially). The one exception is Mem-T, whose trained memory policy is its released post-trained 4B checkpoint. The embedding model is Qwen3-Embedding-4B throughout. A Qwen3-4B-Instruct ablation is in Appendix F.

4 Experimental Results

4.1 Overview

We report performance across benchmarks in Table 3 and per-method preprocessing vs. inference cost in Fig. 1. The following high-level patterns emerge:

(1) **No method dominates every column.** Every index-based method (one that pre-builds

a structured store such as a graph, summary notes, or multi-store cache) lags long context on at least one benchmark. AutoMEM holds the best overall rank (2.75); DCI-Lite (5.20) is among the strongest pre-existing baselines. The shortfall of index-based methods is most visible on agentic-trajectory QA (§4.2).

(2) **Long context is a competitive reference in this suite** (Hu et al., 2026b; Zhao et al., 2026): heavy-index methods such as HippoRAGv2 cannot recover preprocessing cost unless many future queries hit the same store, while lighter alternatives such as LightMem keep the build cheap but rank below long context overall.

(3) **On dynamic agentic tasks, most methods converge** within sampling variance.

(4) **Token efficiency $\neq$ system efficiency.** HippoRAGv2 and AMA-Agent issue tokens as long chains of small serial LLM calls with KV-cache misses, inflating wall-clock and GPU contention far beyond what raw token counts suggest. This reflects a memory system's infrastructure affinity, which token counts alone do not capture; per-method numbers are in Appendix G.

The rest of §4 follows one thread: when to commit memory structure at build time versus query time. §4.2 diagnoses why early commitment fails on agentic QA; §4.3 shows unindexed raw-text search wins by deferring that commitment to query time; §4.4 characterizes when the build-time cost still pays off; §4.5 closes with the scope limit: on dynamic agentic tasks, no memory design closes the gap.

4.2 Where index-based memory fails on agentic QA

Indexing-based methods perform poorly on agentic QA (Table 3, AMABench columns), with several scoring less than half the long-context baseline despite building richer structures. To localize the failure, we ablate the pipeline and ask whether the answer is still recoverable before the normal retrieval interface is applied. HippoRAGv2 extracts an entity-fact KG; AMA-Agent attaches per-turn state reports to a turn-indexed causality graph with action-labeled edges; PlugMem keeps three parallel stores (episodic raw trace, semantic entity facts, procedural subgoal summaries) behind a mode-routing retriever. We sample 60 questions across the three AMABench domains and use Codex with gpt-5.5 (extra high reasoning) to check this oracle storage answerability. Methods designed for personal-chat or Wikipedia-style assistants hit a hard representational ceiling (Table 4); AMA-Agent and PlugMem show a different pattern: the answer is often still present somewhere in storage, but the deployed retrieval interface fails to surface it reliably.

A deeper investigation reveals two failure modes. The first is a **representation-level** failure (Figure 2): HippoRAGv2's step indices and action labels never enter the entity-relation KG because they are not entity-relation facts, so the data is gone before retrieval and no answerer can recover it; the only fix is to widen the schema at build time.

The second is a **retrieval-level** failure: storage is not where the answer is lost. Table 4 shows that PlugMem keeps over 98% of the answer information (its episodic store holds the dense raw trace), so an oracle reading the whole memory could answer almost any question. Its retrieval interface is **single-shot**: a fixed router selects one branch by question-embedding similarity and returns a compact note, without allowing the answerer to request more evidence. When the answer lives in step-level detail, the compact note omits it even though the raw trace retains it. An agent-controlled interface can address this measured failure mode by issuing a follow-up retrieval when the first evidence is insufficient.

A further issue is *inflexibility*: index-based memory commits to a schema at build time, so the designer must anticipate which relation types future questions will probe. To accommodate agentic trajectories, AMA-Agent and PlugMem must add more complexity to the graph and include more node and relation types, which is unlikely to scale as agentic scenarios grow more diverse. Letting an agent actively write the query partially addresses this; the intrinsic problems of index-based memory for agentic use cases are left for future study.

4.3 When unindexed raw-text search wins

Both failures in §4.2 stem from committing memory structure at build time. We now turn to DCI-Lite, one of the strongest pre-existing baselines by mean rank, an unindexed search agent that keeps evidence in raw text and lets the LLM compose a per-question retrieval plan via `grep` and `read`. Compared to index-based methods that commit a schema at build time and pay for that choice with the inflexibility of §4.2, the harness inverts the bind-

Method	LoCoMo *LLM judge*	A-ALF *LLM judge*	A-Web *LLM judge*	A-SQL *LLM judge*	HQA *LLM judge*	AR *LLM judge*	TTL *LLM judge*	LRU *LLM judge*	CR *LLM judge*	ALFW *SR*	MA-S *PS*	MA-T *c-sPS*	LME-S *LLM judge*	Gen. rank
Long context	61.5	26.9	**43.9**	**35.4**	58.0	**82.0**	52.0	6.0	12.0	**46.7**	32.8	12.8	48.0	5.35
SimpleMem	56.5	8.3	18.2	13.8	70.5	56.0	46.0	**26.0**	20.0	43.3	26.7	17.7	13.0	9.20
LightMem	62.0	9.4	25.3	15.6	78.5	62.0	60.0	10.0	16.0	43.3	**38.3**	15.5	49.0	6.40
HippoRAGv2	58.0	10.9	28.6	23.0	83.5	74.0	58.0	10.0	14.0	40.0	31.7	**18.8**	52.0	5.35
PlugMem	53.0	25.0	16.7	18.1	63.5	64.0	58.0	10.0	**24.0**	33.3	28.3	17.7	46.0	9.00
AMA-Agent	54.5	10.0	31.0	22.9	78.0	66.0	42.0	6.0	12.0	31.7	30.0	18.0	42.0	8.35
DCI-Lite	45.0	**32.2**	37.2	25.5	87.5	60.0	54.0	10.0	14.0	41.7	33.3	**18.8**	17.0	5.20
DCI-Lite+Sum	46.5	30.8	38.1	27.2	**89.0**	58.0	48.0	6.0	16.0	40.0	30.6	18.4	15.0	5.95
Mem-T	66.5	26.0	32.1	10.0	63.5	38.0	**66.0**	2.0	10.0	38.3	30.0	18.4	52.0	7.50
MemRL	47.5	5.4	29.9	18.8	62.0	64.0	54.0	4.0	6.0	36.7	33.3	14.4	32.0	9.60
Mem0	53.5	21.7	18.8	22.4	76.5	46.0	42.0	2.0	12.0	33.3	28.3	14.4	**59.0**	9.15
MemoryOS	54.0	29.6	32.8	24.6	66.0	56.0	54.0	4.0	4.0	36.7	33.3	17.7	38.0	7.20
AutoMEM	**67.5**	**32.2**	37.2	26.4	86.5	66.0	62.0	10.0	14.0	41.7	33.3	**18.8**	57.0	**2.75**
Variance	1.4	1.7	0.2	0.6	1.0	2.8	3.0	1.8	2.0	3.9	4.2	1.1	5.0	–

Table 3: Main results: per-method accuracy across all benchmarks; best per column in **bold**. A-ALF/A-Web/A-SQL are AMABench domains; HQA is HotpotQA; AR/TTL/LRU/CR are MemoryAgentBench (Hu et al., 2026b) subtasks; ALFW is ALFWorld; MA-S/MA-T are MemoryArena shopping/travel. The Gen. column is the mean fractional rank across benchmarks, with MemoryAgentBench counted once by averaging AR/TTL/LRU/CR. The Variance row reports across-seed variation in percentage points where available.

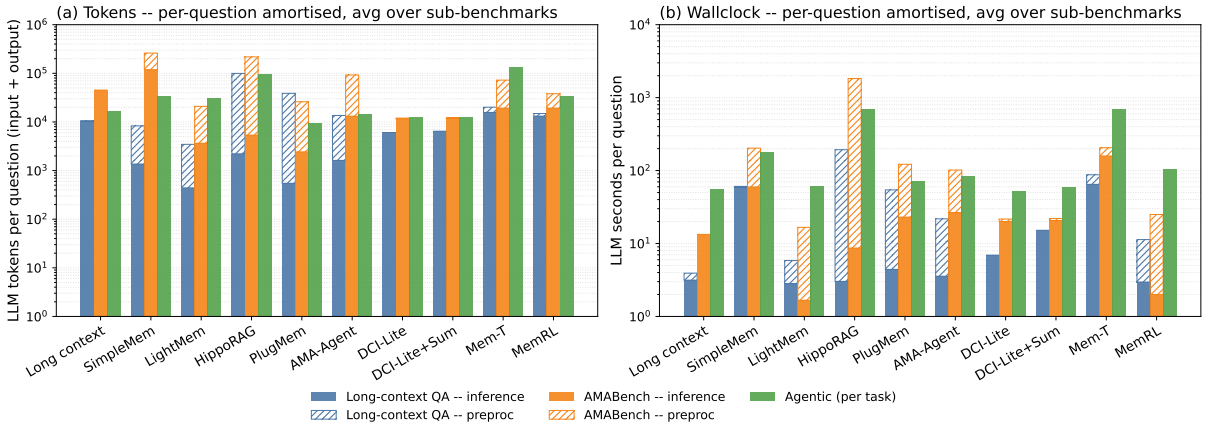


Figure 1: **Per-method preprocessing vs. inference cost per question**, averaged across sub-benchmarks within each task category.

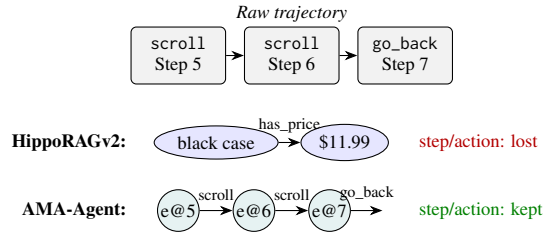


Figure 2: Schema loss on an AMABench-Web question (*"which step performed the return from the product page, and how many scrolls happened before it?"*). HippoRAGv2 keeps entity-relation facts but discards step indices and actions; AMA-Agent's turn-indexed graph keeps both.

Method	A	B	C	D	Overall
HippoRAGv2	19	7	0	14	10
SimpleMem	38	40	33	29	35
AMA-Agent	100	80	67	86	83
PlugMem	100	100	93	100	98

Table 4: Oracle storage answerability (%). The probe asks whether the answer is still recoverable from the method's stored representation, or from the raw episodic trace when the method keeps one; it is therefore an upper bound on what the retrieval interface can deliver, not an end-to-end retrieval score. Question types are AMABench's: A state-value lookup, B causal inference, C causal-chain, D constraint comparison.

ing order: the LLM sees the actual question first, then decides at runtime which slice of raw evidence to read and how to aggregate it. Runtime overhead is comparable to LongCtx since both route the retrieved evidence through an answer LLM; what DCI-Lite buys with its grep–read loop is generality across regimes, not raw token savings on any single benchmark. The cost win is clearer against heavy-index methods: HippoRAGv2 pays a ∼130K-per-question amortized build on long-context QA versus DCI-Lite's near-zero build (Fig. 1). AutoMEM builds on this observation by giving the same controller access to indexed views (§5).

Table 5 reports three signals behind DCI's lead,

Probe (DCI vs. baseline)	Δ pp
Predicate complexity (vs. HippoRAGv2, HQA)	
neither key entity directly in question	+11.3
both key entities directly in question	−3.1
Corpus–window regime (vs. LongCtx)	
LoCoMo, ∼26k tokens (fits window)	−16.5
HotpotQA, 9,811 passages (exceeds window)	+29.5
Schema expressiveness (vs. HippoRAGv2)	
AMABench-ALF, 10% encodable	+21.3

Table 5: Three controlled probes for DCI's generality lead. Δ is DCI minus the named baseline.

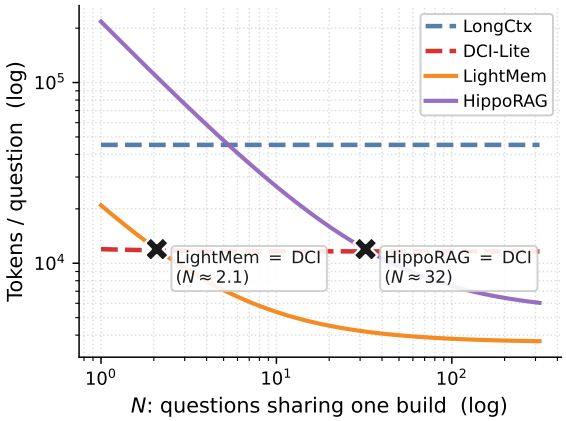


Figure 3: Per-question token cost vs. amortisation N on AMABench. LightMem crosses DCI-Lite at $N{\approx}2$; HippoRAGv2 at $N{\approx}32$.

with one probe per signal while pinning the other two. **Predicate complexity:** HippoRAGv2 indexes Wikipedia entities by name, and each HotpotQA question hinges on two such entities. When the question names both entities, the index lookup is trivial; when it refers to them indirectly, the index must first recover the entity names before it can do anything. Structured indexing tends to win in the first case, DCI in the second. **Corpus–window regime:** the raw-text search agent becomes valuable when a single read of the corpus is impractical. **Schema expressiveness:** the index has a hard representational ceiling that no amount of retrieval quality can lift.

4.4 When indexing pays off

We now study when an index-based method is still worth its build cost, along two axes. From the **efficiency** side, we ask when the per-question token cost of an index drops below the no-build baselines once enough queries amortize the build. From the **performance** side, we ask which task properties make index-based methods a good fit in the first place. The combined picture is that an index pays off only when both axes are satisfied: the workload amortizes the build, *and* the task structure rewards build-time aggregation.

Efficiency axis: the amortization effect. An index-based method pays its build cost once and amortizes it across all questions that hit the same store. As the number of shared questions N grows, its per-question cost drops; no-build methods (LongCtx, DCI-Lite) stay flat. Fig. 3 makes this concrete on AMABench: LightMem crosses DCI-Lite at $N{\approx}2$, and even HippoRAGv2, despite its much larger build, catches up at $N{\approx}32$.

Performance axis: when does indexing help? Being cheap is not the same as being useful: Table 3 shows index-based methods are dominated in accuracy on most benchmarks, even where the build cost is amortized away by enough queries. We first characterize what an index does at storage time, then identify the workload property that makes that work valuable. The index-based methods in our suite come in two flavors. *Graph-based* methods like HippoRAGv2 encode *explicit* relations between stored units (entity-fact triples, turn-indexed causal edges, or multi-store mode tags) at build time. *Semantic note-based* methods like SimpleMem compress multiple turns into LLM-summarized notes and embed the notes, encoding cross-unit content *implicitly* in the summary text. The two flavors look different at the storage layer, but they share one defining property: both pay a build-time cost to **aggregate information across raw units**, so a downstream query can answer with a single retrieval rather than reconstruct the relation on the fly. This shared property, not the specific mechanism, controls when indexing pays off.

Aggregation only pays off when the workload demands it. We measure that demand by the **structural rate**: the fraction of questions whose answer cannot be recovered from any single retrieval unit (passage, dialog turn, or trajectory step) under flat retrieval (grep, BM25, or dense top-5), and therefore requires combining content across units. It is analogous to a retrieval answerability diagnostic at a fixed cutoff, but differs from standard recall@k and QA accuracy: it labels a question–corpus pair's need for cross-unit composition rather than whether a particular method retrieved or answered correctly. It therefore estimates the opportunity for build-time

aggregation rather than an end-to-end system score.

Benchmark	Struct. rate	Avg. Δ vs. DCI-Lite (pp) graph	semantic
LoCoMo	**60.7%**	**+13.0**	**+14.2**
HotpotQA	17.7%	−15.4	−13.0
AMABench-ALF	10.3%	−14.2	−23.4
AMABench-T2SQL	8.6%	− 7.0	−10.8

Table 6: Structural rate is associated with the head-to-head between indexed methods and the unindexed search agent DCI-Lite in the evaluated workloads. Accuracy Δ in pp, averaged over methods in each flavor (graph: HippoRAGv2, PlugMem, AMA-Agent, MemT; semantic: LightMem, SimpleMem).

Observation and implication. Two corpus-side properties jointly gate when indexing pays off, and they play distinct roles. *Schema expressiveness* (§4.3) gates whether indexing *can* help at all: if the schema cannot encode the predicates the questions probe, no amount of structural demand translates into accuracy gains (the AMABench-ALF failure mode). *Structural rate* (this section) gates whether indexing is *needed*: even with a fitting schema, indexing only adds value when many questions require cross-unit aggregation a flat retriever cannot do. Both must hold for indexed methods to beat the unindexed search baseline.

4.5 Scope: dynamic agentic tasks are policy-limited

The preceding subsections assume static-style QA, where the right memory design can improve accuracy. On dynamic agentic tasks this assumption breaks: no memory choice in our suite closes the gap, regardless of when it commits. On ALFWorld, no memory method consistently improves the long-context baseline, while parameter-level post-training (GRPO (Shao et al., 2024) on Qwen2.5-7B) does. To localize where memory is blocked, we hand-craft a **golden procedural memory** by distilling the rules that win ALFWorld games from a small set of expert trajectories into a single static rule sheet injected via the system prompt, and compare four conditions (Table 7). The oracle closes only part of the gap to the trained actor.

The gap splits into two pieces: one that the golden memory can close, and one that it cannot.

Gap 1: over-entropic action posterior. Without a rule in context, the actor wanders across too many candidate actions and runs out of the 50-step budget (all 32 failed trajectories hit the cap). Golden memory fixes this by spelling out the recipe for each task family, and the lift concentrates on tasks whose solution can be written down as a short procedure. Memory systems fail to deliver the same lift because reflections written from past trajectories capture episode-specific events rather than reusable procedures. An expert is needed to distill these memories from trajectories.

Condition (Qwen2.5-7B actor)	ALFWorld SR
Vanilla long-context (no memory)	46.7%
Best memory†	43.3%
Golden procedural memory	71.7%
GRPO post-trained	86.7%

Table 7: ALFWorld success rate with Qwen2.5-7B as the actor backbone. †Strongest of all memory methods on the ALFWorld column of Table 3 (SimpleMem / LightMem at 43.3%).

Gap 2: state-dependent decisions text rules cannot encode. The remaining 17 of 32 failures persist even with the golden recipe in context. Inspecting them, the correct next action depends on the current environment state (e.g., which receptacles are currently open, what the agent is already holding) rather than on a fixed recipe step. A text rule cannot enumerate the right branch for every possible state, so the actor still picks the wrong action at these decision points. The rule can also actively mislead: when its vocabulary surface-matches the task description, the actor mechanically follows it and triggers unnecessary action sequences. Closing this gap requires reshaping the actor's behavior at exactly those state-conditional decision points.

These two gaps connect to two open lines on memory-based agent self-improvement. **Gap 1** explains why an agent struggles to self-summarize its own procedural memory: a weaker actor cannot reliably distill canonical recipes from its own trajectories, so the propositional content must be sourced from a higher-capability LLM oracle and distilled into the actor (as in SkillRL (Xia et al., 2026)) before it can produce the lift. **Gap 2** delineates the capability ceiling of self-evolving agents whose only update channel is non-parametric memory: such agents cannot move past behavior already implicit in the actor's prior. Going beyond this ceiling requires parametric updates such as the RL-based self-evolution in OpenClaw-RL (Wang et al., 2026).

Mem-T and MemRL sit on the non-parametric side: they post-train a memory controller on top of an unchanged actor, so they remain bounded by Gap 2.

5 AutoMEM: A Design Inspired by Our Findings

The experiments in §4 suggest that the unindexed search agent (DCI-Lite) performs well on most benchmarks and only loses ground when an answer must be pieced together from many records, where an index can close the gap if its schema matches the questions. We therefore frame AutoMEM as a **diagnostic-driven strong baseline** rather than a new standalone memory store. It retains DCI-Lite's unindexed raw-text store and `grep`/`read` loop, augmenting it with agent-controlled selection over heterogeneous memory views and view-specific query generation via graph-based indexed queries (`index_query`).

We instantiate this as an agentic loop (Algorithm 1). At each step, the agent (1) picks one action from a four-item menu: `rg` for regex over raw records, `read` for fetching a record by identifier, `dump` for sending a short corpus to the answer LLM, and `index_query` for a typed query against a pre-built graph index. (2) A planner LLM writes the call, an executor runs it, and the evidence is appended to a running trace. (3) An LLM judge inspects the trace; on sufficiency, an answerer composes the final answer, on insufficiency, the judge returns a hint naming what is missing, and the planner is re-invoked. One LLM backbone serves all roles. By default (*LightMem-graph*), `index_query` executes structured Cypher queries (Francis et al., 2018) against a typed property graph induced from LightMem's index: its write-time semantic summaries become nodes connected by relational edges (`SAID_BY`, `OF_TOPIC`, `FOLLOWS`). Unless noted otherwise, AutoMEM results use this variant. An alternative variant (*PlugMem-graph*) uses PlugMem's tag, semantic, and episodic stores with edges `HAS_TAG`, `MENTIONS`, and `SHARES_TAG`.

Results and `index_query` ablation. AutoMEM achieves the best mean rank in Table 3 (2.75, versus 5.20 for DCI-Lite). The largest gains over DCI-Lite occur on the two multi-session benchmarks, LoCoMo and LongMemEval-S. On LongMemEval-S, AutoMEM scores 57.0, just below Mem0's 59.0 and above long context's 48.0. It ties DCI-Lite on seven settings, improves on five, and is 1.0 point lower on HotpotQA, within its measured run-to-run variance.

Algorithm 1 AutoMEM: agentic memory-access loop

Require: memory store M, question q, step budget $K_{\max}$
Require: action set $\mathcal{A} = \{$`rg`, `read`, `dump`, `index_query`$\}$
// trace: evidence so far. hint: judge feedback.
$trace \leftarrow [\,]$; $hint \leftarrow \varnothing$
for $k = 1, \ldots, K_{\max}$ **do**
  // Plan: 1 LLM call, emits one action $\in \mathcal{A}$.
  $a_k \leftarrow \textsc{Plan}(q, trace, hint)$
  // Execute: tool call on M, no LLM.
  $o_k \leftarrow \textsc{Execute}(M, a_k)$
  $trace \leftarrow trace \parallel (a_k, o_k)$
  // Judge: 1 LLM call. Skipped when $K_{\max}$=1 (single-pass).
  $v \leftarrow \textsc{Judge}(q, trace)$
  if $v.sufficient$ **then break**
  else $hint \leftarrow v.reason$
  end if
end for
// Answer: 1 LLM call composes the final answer.
return $\textsc{Answer}(q, trace)$

Removing `index_query` removes the indexed views and leaves the raw-text `grep`/`read` loop, reducing AutoMEM to DCI-Lite. Thus, the ablation tests the bundle of indexed-view availability, agent-controlled view selection, and view-specific query generation; it does not isolate those three elements from one another. Table 8 summarizes the largest contrasts, and Appendix D reports all 13 settings.

Method	LoCoMo	HQA	MAB	LME-S	Rank ↓
AutoMEM	**67.5**	86.5	**38.0**	**57.0**	**2.75**
− `index_query`	45.0	**87.5**	34.5	17.0	5.20
Long context	61.5	58.0	**38.0**	48.0	5.35

Table 8: AutoMEM ablation on representative benchmarks. MAB is the mean over the four MemoryAgentBench subtasks; Rank is the mean fractional rank defined in Table 3. The no-index row is equivalent to DCI-Lite.

Workload-level analysis across 13 settings. Inspecting the full 13-setting results (Table 13) reveals a structural dichotomy in memory utility directly tied to the query workload. On *high structural rate workloads*, most prominently LoCoMo (+22.5 pp, from 45.0 to 67.5) and LongMemEval-S (+40.0 pp, from 17.0 to 57.0), alongside MemoryAgentBench AR (+6.0 pp) and TTL (+8.0 pp), graph-based `index_query` provides the decisive advantage. In these multi-session dialogues, key evidence spans non-adjacent turns separated by hundreds of conversational steps without direct lexical co-occurrence. Graph traversal over the relational edges bridges such turns through intermedi-

ate entity and topic nodes, where lexical grep fails. Conversely, on *sequential trajectories and procedural tasks* across AMABench (ALF: 32.2 vs 32.2, Web: 37.2 vs 37.2, SQL: 26.4 vs 25.5) and embodied environments (ALFW: 41.7, MA-Shop: 33.3, MA-Travel: 18.8), `index_query` yields negligible change. Sequential action-observation logs feature fine-grained temporal state transitions where static schemas inject noise without useful multi-hop edges. AutoMEM's controller dynamically defaults to raw-text `rg`/`read` on procedural logs, avoiding the severe collapse suffered by purely graph-based memories like HippoRAGv2.

General algorithm and feedback loop. Algorithm 1 defines the canonical AutoMEM design: a multi-step plan, execute, and judge loop that re-queries memory when the trace is insufficient. Beyond long context's single call, the loop issues three to four calls per question (planner, optional dump, judge, answerer). If needed evidence is missing, the judge emits a targeted textual hint that guides subsequent planning, directly resolving the needle-in-a-haystack failures common in passive single-shot retrievers.

Systems efficiency and deployment trade-offs. On top of Algorithm 1, we examine two latency optimizations that retain accuracy at a fraction of token cost (Table 9). First, for *KV-cache-friendly prompt structure*, standard agent loops invalidate caches across turns. Placing invariant instructions, signatures, and schema hints at the prompt head while keeping dynamic traces at the tail increases Qwen3-32B prefix-cache hit rates under vLLM / SGLang from 5.4% to over 50%, resolving multi-call overhead to fast cache reads. Second, as an *architectural single-pass parallel cap*, fixing $K_{\max}$=1 and dispatching `rg`, `read`, and `index_query` concurrently cuts query tokens by 66% to 72% across LoCoMo (24.8K $\rightarrow$ 6.9K), HotpotQA (5.9K $\rightarrow$ 2.0K), and AMABench (25.3K $\rightarrow$ 8.2K), with a modest 4.0 pp penalty on LoCoMo (63.5 vs 67.5). At 6.9K tokens, single-pass AutoMEM is cheaper than long context (19.8K) while remaining +2.0 pp more accurate (63.5 vs 61.5).

Evaluation reliability and judge audit. To verify that findings are not artifacts of evaluation bias, we audit the judge rubric across models and human annotators (Appendix C). Evaluating four representative systems on LoCoMo under four LLM judges (Qwen3-32B, Claude-Sonnet-4.5, DeepSeek-V3, GLM-4.7) shows that all four preserve the exact relative ranking (AutoMEM > LightMem > SimpleMem > DCI-Lite; Table 12). Furthermore, a blinded human audit on 60 diagnostic trajectory questions yields a 91.7% agreement rate (55/60) with the primary judge, confirming that the cross-scenario performance separations reflect substantive memory differences.

	LoCoMo		HotpotQA		AMABench	
Method	build	query	build	query	build	query
Long context	0K	19.8K	0K	0.9K	0K	32.8K
DCI-Lite	0K	10.8K	0K	1.3K	0K	8.8K
HippoRAGv2	13.5K	3.5K	181.1K	0.9K	163.1K	12.8K
AutoMEM (full loop)	5.5K†	24.8K	0.6K†	5.9K	20.8K†	25.3K
AutoMEM (single-pass)	5.5K†	6.9K	0.6K†	2.0K	20.8K†	8.2K

Table 9: Per-question LLM token cost; preprocessing is amortised the same way as Fig. 1. †AutoMEM's build equals LightMem-graph's preprocessing; the PlugMem-graph variant substitutes PlugMem's (24.2K, 52.2K, 41.5K).

6 Conclusion

We revisit memory systems for LLM agents through a cross-scenario generality and cost lens. Our evaluation quantifies three key observations: (1) no single memory method excels across every task family, while DCI-Lite is among the strongest pre-existing baselines; (2) on audited agentic trajectories, build-time schemas can omit vital step- and action-level evidence, while single-shot retrieval interfaces fail to surface stored evidence; and (3) combining agent-controlled raw-text retrieval with selective indexed access yields the best mean rank in our suite. AutoMEM instantiates this third observation as a diagnostic-driven strong baseline rather than a static memory store.

Two directions appear especially promising for future work: (i) *a memory DSL that automates schema design*: replacing rigid Cypher formulations with domain-specific memory primitives would enable schemas to be meta-learned dynamically per workload (Xiong et al., 2026); and (ii) *dedicated serving infrastructure for memory-bound agents*: as serial prefill across multi-turn plan, judge, and answer steps dominates wall-clock time (§G), program-aware serving stacks that batch intermediate calls and maximize prefix-cache reuse (Kang et al., 2026) can close the remaining latency gap relative to long-context processing.

Limitations

No tiered budget routing. AutoMEM runs plan, judge and answer composition on the same backbone, but the three steps differ in difficulty: planning over a small corpus catalog and judging a candidate answer's sufficiency are short, low-entropy decisions, while answer composition is the step that needs the strong model's reasoning. A per-step router that sends plan and judge to a cheap tier and reserves the strong backbone for answer composition could reduce AutoMEM's per-question LLM cost. We do not implement or measure this router and report uniform-backbone numbers throughout, so the reported cost of AutoMEM should be read as an upper bound.

Ethical Considerations

Data and licensing. All datasets used in this work (HotpotQA, LoCoMo, LongMemEval-S, MemoryAgentBench, AMABench, ALFWorld, MemoryArena) are publicly released research benchmarks under their original licenses, and we use them only for evaluation. We do not collect new human-subject data and we do not redistribute the underlying corpora; only derived metrics, judged-answer files, and run configurations are released through our HuggingFace artifact repository.

Models. Our backbones (`Qwen3-32B`, `Qwen3-Embedding-4B`, `Qwen2.5-7B`, `Qwen3-4B-Instruct`) are open-weight models released by their authors under permissive licenses. All results are from inference-time use through standard serving stacks (sglang, vLLM).

Risks and intended use. This paper studies the cost and generality of memory systems for LLM agents in benchmark settings. It does not deploy agents in user-facing products and does not store, retrieve, or operate on personal data. The agentic memory recipe we recommend, AutoMEM, gives the LLM tool-calling control over a memory store. In a deployment setting where the memory contains user data, the same surface would need standard safeguards (access control, content filtering, audit logging) that are out of scope for this paper.

Compute and environmental cost. Our experiment runs on a mix of GPUs (NVIDIA B200, H200, RTX PRO 6000). We report per-question token and wallclock costs in the main paper so that practitioners can estimate compute before reproducing. All cost analysis is conducted on an isolated B200 instance.

Acknowledgments

Jialiang Gu, Junyu Yin, and Keren Zhou acknowledge support from the U.S. National Science Foundation under Award No. 2411134 and from NVIDIA. Zhikai Chen, Xianxuan Long, Shenglai Zeng, Kai Guo, and Jiliang Tang acknowledge support from the National Science Foundation (NSF) under grant numbers CNS2321416, IIS2212032, IIS2212144, IIS2504089, DUE2234015, CNS2246050, DRL2405483, and IOS2035472, and from the Michigan Department of Agriculture and Rural Development, the U.S. Department of Commerce, the Gates Foundation, Amazon Faculty Award, Meta, NVIDIA, Microsoft, and SNAP. We thank the authors of the systems included in this release: MemoryAgentBench, MemoryOS, A-Mem, LightMem, MemRL, Mem-T, and PlugMem.

References

Qingyao Ai, Yichen Tang, Changyue Wang, Jianming Long, Weihang Su, and Yiqun Liu. 2025. MemoryBench: A benchmark for memory and continual learning in LLM systems. *ArXiv preprint*, abs/2510.17281.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready AI agents with scalable long-term memory. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *ArXiv preprint*, abs/2404.16130.

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. 2025. LightMem: Lightweight

and efficient memory-augmented generation. *ArXiv preprint*, abs/2510.18866.

Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An evolving query language for property graphs. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*, pages 1433–1445.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*.

Zexue He, Yu Wang, Churan Zhi, Yuanzhe Hu, Tzu-Ping Chen, Lang Yin, Ze Chen, Tong Arthur Wu, Siru Ouyang, Zihan Wang, Jiaxin Pei, Julian McAuley, Yejin Choi, and Alex Pentland. 2026. MemoryArena: Benchmarking agent memory in interdependent multi-session agentic tasks. *ArXiv preprint*, abs/2602.16313.

Stefan Heule, Emily Jia, and Naman Jain. 2025. Improving agent with semantic search. Cursor Blog. Published November 6, 2025.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*.

Tiansheng Hu, Yilun Zhao, Canyu Zhang, Arman Cohan, and Chen Zhao. 2026a. SAGE: Benchmarking and improving retrieval for deep research agents. *ArXiv preprint*, abs/2602.05975.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2026b. Evaluating memory in LLM agents via incremental multi-turn interactions. In *International Conference on Learning Representations (ICLR)*.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, et al. 2025. Memory in the age of AI agents. *ArXiv preprint*, abs/2512.13564.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*.

Hao Kang, Ziyang Li, Weili Xu, Xinyu Yang, Yinfang Chen, Junxiong Wang, Beidi Chen, Tushar Krishna, Chenfeng Xu, and Simran Arora. 2026. ThunderAgent: A simple, fast and program-aware agentic inference system. *ArXiv preprint*, abs/2602.13692.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory OS of AI agent. *ArXiv preprint*, abs/2506.06326.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. 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, Online. Association for Computational Linguistics.

Dong-Ho Lee, Adyasha Maharana, Jay Pujara, Xiang Ren, and Francesco Barbieri. 2025. REALTALK: A 21-day real-world dataset for long-term conversation. *ArXiv preprint*, abs/2502.13270.

Patrick S. H. 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. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, Chengbao Liu, and Jie Tan. 2026a. TiMem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. *ArXiv preprint*, abs/2601.02845.

Yifei Li, Weidong Guo, Lingling Zhang, Rongman Xu, Muye Huang, Hui Liu, Lijiao Xu, Yu Xu, and Jun Liu. 2026b. Locomo-plus: Beyond-factual cognitive memory evaluation framework for LLM agents. *ArXiv preprint*, abs/2602.10715.

Zhuofeng Li, Haoxiang Zhang, Cong Wei, Pan Lu, et al. 2026c. Beyond semantic similarity: Rethinking retrieval for agentic search via direct corpus interaction. *ArXiv preprint*, abs/2605.05242.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026a. SimpleMem: Efficient lifelong memory for LLM agents. *ArXiv preprint*, abs/2601.02553.

Xiaoyuan Liu, Tian Liang, Dongyang Ma, Deyu Zhou, Haitao Mi, Pinjia He, and Yan Wang. 2026b. The pensieve paradigm: Stateful language models mastering their own context. *ArXiv preprint*, abs/2602.12108.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe.

2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. 2022. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Natchanon Pollertlam and Witchayut Kornsuwannawit. 2026. Beyond the context window: A cost-performance analysis of fact-based memory vs. long-context LLMs for persistent agents. *ArXiv preprint*, abs/2603.04814.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. 2025. MemoRAG: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM Web Conference 2025 (WWW)*.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024. LaMP: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7370–7392. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *ArXiv preprint*, abs/2402.03300.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR)*.

Yiheng Shu, Saisri Padmaja Jonnalagedda, Xiang Gao, Bernal Jiménez Gutiérrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. 2026. REMem: Reasoning with episodic memory in language agent. *ArXiv preprint*, abs/2602.13530.

Anxin Tian, Yiming Li, Xing Li, Hui-Ling Zhen, Lei Chen, Xianzhi Yu, Zhenhua Dong, and Mingxuan Yuan. 2026. SwiftMem: Fast agentic memory via query-aware indexing. *ArXiv preprint*, abs/2601.08160.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. 2026. OpenClaw-RL: Train any agent simply by talking. *ArXiv preprint*, abs/2603.10165.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. 2025. Mem-α: Learning memory construction via reinforcement learning. *ArXiv preprint*, abs/2509.25911.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations (ICLR)*.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. 2026. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning. *ArXiv preprint*, abs/2602.08234.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. TravelPlanner: A benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Yiming Xiong, Shengran Hu, and Jeff Clune. 2026. Learning to continually learn via meta-learning agentic memory designs. *ArXiv preprint*, abs/2602.07755.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Xiucheng Xu, Bingbing Xu, Xueyun Tian, Zihe Huang, Rongxin Chen, Yunfan Li, and Huawei Shen. 2026. Chain-of-memory: Lightweight memory construction with dynamic evolution for LLM agents. *ArXiv preprint*, abs/2601.14287.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. 2025. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *ArXiv preprint*, abs/2508.19828.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *ArXiv preprint*, abs/2505.09388.

Ke Yang, Zixi Chen, Xuan He, Jize Jiang, Michel Galley, Chenglong Wang, Jianfeng Gao, Jiawei Han, and ChengXiang Zhai. 2026. PlugMem: A task-agnostic plugin memory module for LLM agents. *ArXiv preprint*, abs/2603.03296.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. 2026. MemAgent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *International Conference on Learning Representations (ICLR)*.

Yanwei Yue, Guibin Zhang, Boci Peng, Xuanbo Fan, Jiaxin Guo, Qiankun Li, and Yan Zhang. 2026. Mem-T: Densifying rewards for long-horizon memory agents. *ArXiv preprint*, abs/2601.23014.

Haozhen Zhang, Haodong Yue, Tao Feng, Quanyu Long, Jianzhu Bao, Bowen Jin, Weizhi Zhang, Xiao Li, Jiaxuan You, Chengwei Qin, and Wenya Wang. 2026a. Learning query-aware budget-tier routing for runtime agent memory. *ArXiv preprint*, abs/2602.06025.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. 2026b. MemRL: Self-evolving agents via runtime reinforcement learning on episodic memory. *ArXiv preprint*, abs/2601.03192.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, Yuandong Tian, and Jishen Zhao. 2026. AMA-Bench: Evaluating long-horizon memory for agentic applications. *ArXiv preprint*, abs/2602.22769.

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

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. 2026. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *International Conference on Learning Representations (ICLR)*.

A Use of AI Writing Assistance

In line with the policy on AI Writing Assistance, we use AI assistants for code development, language polishing, and grammar of the manuscript. All scientific claims, experimental design, results, and analyses are authored, reviewed, and verified by the human authors.

B Dataset Details and Selection Criteria

Running all methods on the full release of each benchmark would be prohibitively expensive: a single full sweep would cost hundreds of GPU-hours of LLM inference plus tens of millions of judge tokens. We therefore evaluate every method on a fixed, deterministic subset per benchmark, chosen to preserve question-type coverage. Table 10 lists the subset configuration for each task.

Benchmark	Subset / setting
HotpotQA	n=200 on the HippoRAGv2 shared corpus (9,811 passages)
LoCoMo	n=200 questions sampled from the public release
LongMemEval-S	stratified n=100 subset, fixed manifest, seed 42
MemoryAgentBench	200-question subset spanning AR / TTL / LRU / CR sampled uniformly based on question length
AMABench	ALF / Web / T2SQL domains
ALFWorld	60 test tasks in total, sampled uniformly across task types
MemoryArena	n=30 per scenario (shop, travel)

Table 10: Benchmark subsets used in all main experiments. Subsets are fixed across methods and runs to make accuracy and cost numbers directly comparable.

B.1 Why We Use the LLM Judge as the Primary QA Metric

We use the LLM judge rather than token F1 on every QA benchmark because F1 is sensitive to answer verbosity rather than correctness, and LoCoMo is the cleanest illustration in our suite. LoCoMo answers are naturally verbose: questions probe multi-session personal context, so models respond in full sentences rather than entity spans. For a question whose gold answer is “Sweden”, the

prediction "Sweden" scores F1 1.00 while "The country is Sweden" scores 0.50 after standard answer normalization, a 2× gap for the same underlying fact. The effect compounds at the system level: holding the model fixed (long-context GPT-4o-mini) and changing only the answering prompt produces a 34-point swing in average LoCoMo F1, driven entirely by output verbosity rather than retrieval or reasoning quality (Table 11). HotpotQA gold answers are short entity spans ("Eiffel Tower", "1887", "Stephen King"), so the gap is smaller, but it does not vanish. We therefore use the LLM judge uniformly across QA benchmarks.

Answering prompt	LoCoMo F1 (%)
SimpleMem paper prompt	18.7
LoCoMo paper prompt	36.1
Terse-answer prompt	52.9

Table 11: LoCoMo token F1 under three answering prompts, holding the model fixed to long-context GPT-4o-mini.

C Judge Robustness and Human Verification

We re-score the same answer files with the paper's Qwen3-32B judge and three additional judges: Claude-Sonnet-4.5, DeepSeek-V3, and GLM-4.7. Although the absolute LoCoMo scores shift (Table 12), all four judges produce the same ordering. The mean Spearman rank correlation is 1.0, and per-question pairwise agreement is 91.7%. On AMABench, the ordering is likewise unchanged, with 92.5% pairwise agreement and mean rank correlation 1.0.

Method	Qwen	Sonnet	DeepSeek	GLM
AutoMEM	67.5	62.5	63.0	64.0
LightMem	62.0	56.0	57.5	59.0
SimpleMem	56.5	50.5	52.5	54.5
DCI-Lite	45.0	41.0	44.0	46.0

Table 12: LoCoMo accuracy under four LLM judges. All four preserve the method ordering.

We also conduct a blinded human check on the 60-question failure-analysis set, balanced across methods and question types. The annotator grades each prediction against the gold answer without seeing the LLM verdict, labeling 33 predictions incorrect and 27 correct. Human and LLM labels agree on 55 of 60 questions (91.7%). The five disagreements are borderline numeric-rounding or preference-style cases and show no consistent direction of judge bias.

D AutoMEM Indexed-View Ablation

Table 13 gives the complete `index_query` ablation. Removing `index_query` leaves the DCI-Lite raw-text retrieval loop. Indexed access produces its clearest gains on LoCoMo and LongMemEval-S, while several trajectory and environment settings are unchanged.

E Baseline Implementation Details

We run each memory baseline from its official implementation, either vendored into our code repository or installed as the released package, so that results can be reproduced against the upstream code. Two baselines required further changes:

AMA-Agent. The released open-source code has gaps relative to the system described in the original paper (Zhao et al., 2026). We adapt the implementation to match the paper's description.

MemoryArena. The release provides the dataset but not the simulated environment. We implement an approximated environment ourselves and release it alongside the rest of our reproducibility artifacts.

F Results on Other Backbones

Our goal here is to investigate how the quality of the LLM backbone influences our conclusions. We select `Qwen3-4B-Instruct` since Mem-T also adopts this backbone. We find that the instruct model works much better than its thinking counterpart. All other components (the embedding model, the 32B strict factual judge, prompts, subsets, and per-method configs) are held fixed to the main protocol. ALFW, MA-S, and MA-T are omitted because Qwen3-4B falls below the capability threshold required by these benchmarks: every method collapses to ≈ 0 on them, meaning the columns do not differentiate between methods. Based on these results, we find that the effectiveness of DCI-Lite on MemoryAgentBench decreases, which can be attributed to the smaller model's weaker capability to generate appropriate queries. Moreover, we find that for smaller models, index-based memory systems actually hold a greater advantage over long-context methods. One interesting phenomenon we observe is that across a series of tasks, MemRL's performance is actually better with the 4B model

Method	LoCoMo	A-ALF	A-Web	A-SQL	HQA	AR	TTL	LRU	CR	ALFW	MA-S	MA-T	LME-S
AutoMEM	67.5	32.2	37.2	26.4	86.5	66.0	62.0	10.0	14.0	41.7	33.3	18.8	57.0
− `index_query`	45.0	32.2	37.2	25.5	87.5	60.0	54.0	10.0	14.0	41.7	33.3	18.8	17.0

Table 13: AutoMEM with and without indexed-view access across all 13 reported settings.

Method	LoCoMo *LLM judge*	A-ALF *LLM judge*	A-Web *LLM judge*	A-SQL *LLM judge*	HQA *LLM judge*	AR *LLM judge*	TTL *LLM judge*	LRU *LLM judge*	CR *LLM judge*
Long context	58.5	8.9	19.7	21.2	46.5	22.0	48.0	6.0	6.0
SimpleMem	46.0	7.5	0.6	2.1	25.5	8.0	6.0	6.0	2.0
LightMem	63.0	8.1	16.1	16.7	72.0	40.0	58.0	4.0	10.0
HippoRAGv2	61.0	9.4	16.4	17.9	73.0	50.0	44.0	4.0	14.0
PlugMem	59.0	5.6	2.5	4.6	57.5	30.0	14.0	4.0	8.0
AMA-Agent	51.5	10.0	15.0	11.2	72.5	34.0	20.0	4.0	2.0
MemRL	62.0	14.7	24.4	22.9	55.5	30.0	60.0	6.0	10.0
DCI-Lite	50.0	22.8	28.3	14.2	80.5	28.0	20.0	2.0	6.0
DCI-Lite+Sum	48.0	21.7	17.8	14.2	80.5	32.0	18.0	6.0	8.0

Table 14: Qwen3-4B-Instruct-2507 backbone ablation. Mem-T is ignored here since the result in Table 3 is already taken from a post-trained 4B model.

backbone, which may stem from a distinction between thinking and instruction modes. Generally, these results align with our intuition: historically, when LLM capabilities were limited, index-based memory dominated since it could improve both performance and efficiency. However, as LLM capabilities grow, we have crossed an inflection point where memory management paradigms should innovate to become LLM-led. While the Qwen3-32B model used in our main text is a mid-tier LLM, we believe that active management will present even more pronounced advantages for frontier LLMs.

G Token Efficiency vs. System Efficiency

In §3 we track per-question token cost alongside accuracy. However, token counts alone do not capture deployed cost: wall-clock latency, GPU utilization, and KV-cache reuse all depend on how the tokens are issued, not just how many. This appendix unpacks that gap in two steps. First, Table 15 reports serving-stack measurements to complement accuracy and token counts, taken from Qwen3-32B on sglang on a single B200. Second, Table 16 classifies each system-efficiency finding as either an implementation gap that engineering can close or a structural tradeoff tied to the method's design.

Three patterns stand out. First, the methods with the lowest stable-prefix fraction (HippoRAGv2, AMA-Agent, SimpleMem) are exactly the methods where the wall-vs-token gap is largest; the prefix column predicts the gap better than the LLM-calls-per-question column does. Second, the average in-flight requests is $\sim$1 across every structured-memory method we audited, regardless of stated design: the orchestrator dispatches one request per chunk, so continuous batching has nothing to batch against, and decode MFU lands in the lower range. Third, decode MFU correlates negatively with prefill-to-decode ratio: methods that send long prompts and decode short outputs leave the decode kernels idle even when their token totals look low. Long context and DCI-Lite sit at the top of every column because they neither break the prefix nor serialize the orchestration.

The takeaway for future memory-system reports is that per-question token counts can hide order-of-magnitude differences in wallclock and dollar cost. We recommend publishing the columns in Table 15 alongside accuracy and token totals so that the comparison cannot be dominated by orchestrator choices.

Case-by-case: implementation problems vs. inherent tradeoffs. Not every system-efficiency finding is closeable by engineering: some are structurally tied to what the method is designed to do. Table 16 classifies each finding as either an implementation problem (an engineering pass would close most of the gap without changing the method), an inherent tradeoff (closing the gap would require redesigning the method), or a mixed case. The reading: most of HippoRAGv2's gap is closable by async fan-out and prompt-shape changes, AMA-Agent's gap is largely structural (the tool loop and mutating state are the method's premise), and SimpleMem sits in between.

Method	Stable prefix fraction	Avg. in-flight requests	Prefill:decode tokens/call	LLM calls per question	Build wall (HotpotQA)	Decode MFU (%)
Long context	0.85–0.95	∼1	∼1:1	1	0 s	55–70
SimpleMem	0.10–0.25	∼1	∼10:1	1 (+ embed)	∼700 s	25–45
LightMem	0.20–0.40	∼1	∼8:1	1	∼1,400 s	30–50
HippoRAGv2	0.05–0.15	∼1	∼20:1	∼1	12,564 s	20–35
PlugMem	0.10–0.20	∼1	∼15:1	1–2	∼2,600 s	25–40
AMA-Agent	0.05–0.10	∼1	∼100:1	variable (loop)	∼1,500 s[†]	15–30
Mem-T	0.30–0.50	∼1	∼5:1	1 (+ DB ops)	∼900 s	35–55
DCI-Lite	0.70–0.85	∼1	∼2:1	variable (loop)	0 s	50–65

Table 15: Serving-stack metrics per method, Qwen3-32B on sglang on a single B200. HotpotQA's numbers are recorded using a 2000-corpus subset.

Finding	Class	Why
HippoRAGv2 serial build calls	impl.	chunks are independent; async fan-out would batch them.
SimpleMem serial build embeds	impl.	corpus embeddings are independent.
AMA-Agent serial tool loop (query)	tradeoff	each call conditions on prior tool result.
HippoRAGv2 proposition prompts split prefix	impl.	re-template to put chunk in trailing user block.
AMA-Agent `state_mem_str` mutates prefix	tradeoff	evolving state is the method's premise.
SimpleMem rolling-summary rewrite	mixed	append-only is impl.; rewrite is design.
AMA-Agent long prefill, short decode	tradeoff	long state in, short action out is by design.
HippoRAGv2 long build prefill / short output	tradeoff	per-chunk extraction shape is structural.
Embedder co-located on chat GPU	impl.	deploy embedder on CPU or a dedicated GPU.
HippoRAGv2 builds wallclock breakeven	tradeoff	total work scales with C; packing only shifts constants.
Memory store as Python dict in JSON	impl.	switch to RocksDB / LanceDB / Qdrant; no algorithmic change.
Per-question variable call count	tradeoff	data-dependent reasoning loop is part of the method.
Decode MFU loss from prefix-cache miss	mixed	impl. for HippoRAGv2, tradeoff for AMA-Agent.

Table 16: Classification of every system-efficiency finding in this appendix. impl.: closeable by engineering without changing the method. tradeoff: structurally tied to the method's design. mixed: one component closeable, another structural.