ARKA: TYPED MEMORY ACCESS FOR EFFICIENT LONG-CONTEXT INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context LLM agents accumulate dialogue histories, task traces, exemplars, and evolving facts, and full-context prompting exposes this growing state to the backbone LLM, making the KV cache expand with the interaction history. Full-context prompting repeatedly prefills irrelevant history, while prompt compression operates after long-context material has been gathered and typically requires an additional model pass. We propose ARKA, an LLM-free lightweight framework that treats agent memory as query-conditioned working-context assembly. ARKA stores histories and typed memory units outside the backbone LLM, uses a lightweight router to decide how memory should be accessed for each query, assembles compact evidence on CPU, and calls the backbone LLM at most once for generation. Our information-theoretic analysis establishes sufficient conditions for compact working contexts to preserve answer-relevant information and shows that discarding required source relations can prevent answer recovery even when relevant words remain. The LLM-free routing framework is evaluated on Llama-3.1-8B, Qwen3-8B, and Qwen3.5-9B backbones without backbone-specific retraining. Across MemoryAgentBench, LongBench-5, LoCoMo, and MemBench, ARKA outperforms the mBERT-based LLMLingua-2-small compressor in all 11 comparable benchmark–backbone settings, with an average gain of 9.9 points and a median 6.4× speedup in recorded processing time. In a history-growth stress test, full-context prompting runs out of memory at 50× scale on an A100 80 GB GPU, while ARKA reaches 100× with only a few hundred backbone input tokens. These results show that scalable agent memory should be built by learning which memory evidence to activate and which access policy to use for each query, rather than by extending or compressing a monolithic prompt.

1 INTRODUCTION

Long-context LLM agents accumulate conversations, task traces, evolving facts, documents, and exemplars. A personal assistant may retain months of history to answer a question about one appointment. Repeatedly reading this history makes GPU state and inference cost grow with stored information. External memory separates storage from active context (Lewis et al., 2020; Packer et al., 2023; Xu et al., 2025), but the backbone still processes the selected evidence. Prompt compression shortens input through additional scoring or encoding (Pan et al., 2024; Li et al., 2023). Before deciding how much text to expose, we ask what operation the query requires. A latest-fact query needs update order, a short reply needs its preceding exchange, and a demonstration needs its paired label.

We frame this decision as working-context assembly. A memory-access policy chooses the representation to search, the operation to execute, and the evidence budget and format. The same history can therefore support different access paths without being rewritten for each question. Compact access must preserve the source relations needed for answering, alongside relevant text.

Adaptive Routing and Knowledge Assembly (ARKA) implements this access layer over source-linked CPU memory. A learned controller proposes policies, constrained by schema guards and source adapters. Structured paths answer directly, while evidence paths call the unchanged backbone LLM reader once (Figure 1).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

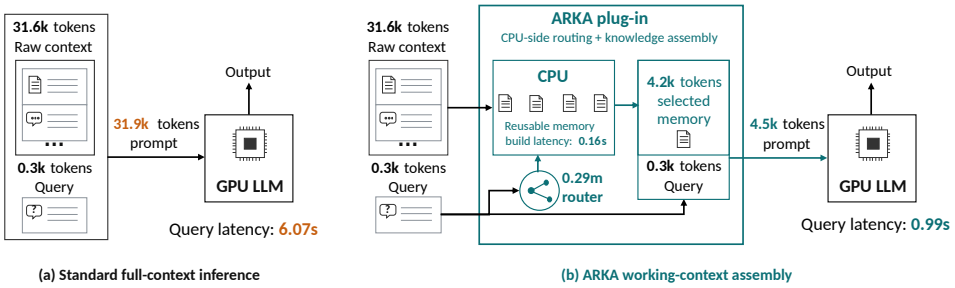


Figure 1: Persistent memory and active reader input. Size and latency annotations illustrate the pipeline, with benchmark results reported in Section 5. The depicted 0.29M router is the first stage of the complete 0.58M-parameter controller.

ARKA targets services that prepare many histories and revisit them intermittently. Reusable CPU views retain the source relations needed for answering without keeping every user's full-history KV state resident.

We summarize our contributions as follows:

- We propose query-conditioned memory access that selects semantic operators and assembles bounded evidence, with direct execution for structured queries. On MemoryAgentBench (MAB), the complete access layer scores 1.5–6.4 points above full context across three backbones using one frozen controller.
- We establish sufficient conditions for compact working contexts to preserve answer information. Our analysis characterizes access-induced information loss and bounds answer recovery when update order is erased, explaining why evidence selection must preserve source relations as well as relevant words.
- We implement reusable, source-linked CPU memory without LLM-based index construction, separating persistent storage from active backbone state. A CPU replay constructs memory views for ten LoCoMo histories in 0.464 s, and the Llama-3.1-8B evaluation reduces mean reader input by 91.5%.

2 Related Work

External memory and evidence organization. RAG and dense retrieval provide reusable access to external knowledge (Lewis et al., 2020; Karpukhin et al., 2020). Retrieval-augmented training can integrate this access into a language model (Guu et al., 2020; Borgeaud et al., 2022; Izacard et al., 2023). Agent memory adds persistent state and memory management: MemGPT uses LLM-managed tiers, MemoryBank maintains long-term interaction memory, ReadAgent forms gist memories, and A-Mem builds linked notes (Packer et al., 2023; Zhong et al., 2024; Lee et al., 2024; Xu et al., 2025). RAPTOR constructs recursive summaries, while HippoRAG uses structured graph recall (Sarthi et al., 2024; Gutiérrez et al., 2024). ARKA's design choice is deterministic, source-linked construction coupled to explicit CPU operators. Its views preserve existing structure without requiring a generative model to author the index.

Adaptive retrieval and compression. Adaptive-RAG selects retrieval depth using downstream-outcome supervision (Jeong et al., 2024). BudgetMem learns query-aware memory budgets with cost-sensitive reinforcement learning (Zhang et al., 2026). ARKA selects semantic operators from prescribed route labels. Prompt compression selects tokens or passages (Jiang et al., 2023; 2024; Pan et al., 2024; Li et al., 2023; Xu et al., 2024). Provence combines pruning with reranking (Chirkova et al., 2025). Both operations can be incorporated into ARKA's evidence-producing paths.

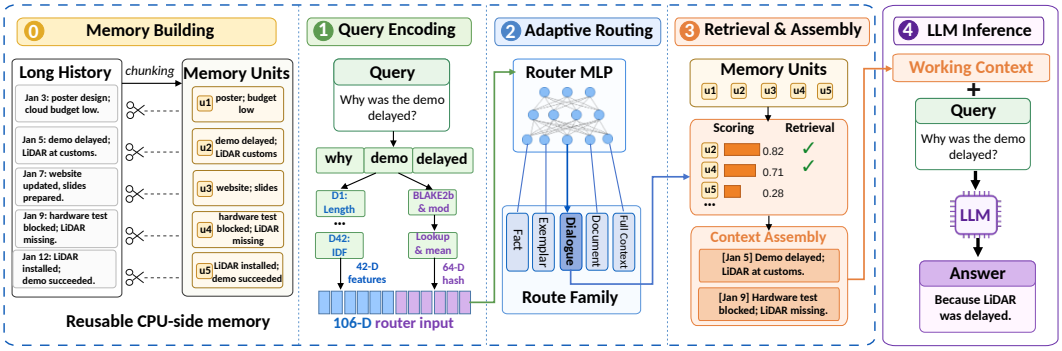


Figure 2: ARKA's evidence-producing path. A reusable CPU index exposes source-linked units for policy selection, evidence assembly, and answer generation. The LiDAR exchange is schematic. Direct paths bypass the reader. Algorithm 1 specifies execution, and Section 3.3 follows a recorded conversation.

3 METHOD

We describe reusable memory construction, query-time dispatch and assembly, then trace a recorded query through the pipeline.

3.1 ARCHIVE, VIEWS, AND EXECUTION CONTRACTS

The history $\mathcal{H} = (x_1, \ldots, x_N)$ is an ordered collection of observed source records: document text, speaker-tagged turns, explicit fact updates, or labeled demonstrations. Each record has a source identifier and any supplied time, speaker, or label metadata. Evaluation answer labels are not used to construct the archive or its views. Builders expose only the views supported by the input structure:

$$\mathcal{M} = \{\mathcal{V}_s : s \in \mathcal{S}(\mathcal{H})\}, \qquad \mathcal{V}_s = B_s(\mathcal{H}). \tag{1}$$

Here $\mathcal{S}(\mathcal{H})$ lists supported view types and B_s builds view $\mathcal{V}_s$. Documents supply cards and chunks, speaker-tagged sessions additionally supply windows, and explicit updates enable versioned records.

For a source unit w_j, a textual view stores a short access representation, its position, and pointers to the original text:

$$u_j = (z_j, \tau_j, m_j), \qquad z_j = [\sigma_j; \mathrm{text}(\mathrm{Core}_m(w_j))]. \tag{2}$$

Here $[a; b]$ denotes concatenation, σ_j is source metadata, τ_j is time or position, and m_j maps retained text to source spans. Core_m selects up to m salient sentences (two to four by policy), and text renders them. Cards are short retrieval handles whose pointers recover omitted source sentences. Dialogue windows index contiguous turns, exemplars retain input–label pairs, and facts retain update order. Builders require no reader calls and are reusable.

An access contract specifies the supported view, preserved relation, and output (Table 1). Textual operators use BM25 scoring (Robertson & Zaragoza, 2009). Cards locate recoverable source spans, windows preserve speaker order, and fact resolution selects the latest eligible record when one record determines the answer. Multiple-choice resolution ranks question–option support directly. Appendix G gives its lexical rules and evidence budgets.

Source recovery. A card locates a promising unit through z_j, then recovers source text through m_j. For a fixed view library with bounded duplication, CPU storage grows linearly with the archive while active reader state depends on the selected prompt and generated answer. Full-context access deliberately exposes more information.

Table 1: Memory views and access operators. Each view preserves the source relations needed by its operator. Evidence budgets are policy parameters, and full reading is bounded by the reader's context limit.

Source structure	Indexed view	Operator and preserved relation	Output
Documents or sessions	Extractive cards and raw chunks	Rank handles, recover neighboring source spans	Capped evidence
Speaker-tagged dialogue	Windows or turn pairs	Rank local exchanges, retain adjacency and speaker order	Capped evidence
Ordered fact updates	Versioned records	Match entity/relation, resolve latest eligible value	Direct answer
Labeled examples	Flat or label-grouped banks	Rank inputs, restore their paired labels	Demonstrations
Options with source evidence	Option-conditioned support	Score question–option support, select an option	Direct answer
General source text	Original history	Read original order and formatting	Full context

3.2 Query-Conditioned Dispatch and Assembly

ARKA predicts a policy from a query representation combining a hashed lexical embedding and 42 numeric features:

$$h(q) = \left[\frac{1}{|\mathcal{G}(q)|} \sum_{a \in \mathcal{G}(q)} E_{\rho(a)} \; ; \; g(q) \right], \qquad \hat{r} = \arg\max_{r} \mathrm{softmax}(\mathrm{MLP}_{\theta_1}(h(q)))_r. \tag{3}$$

$\mathcal{G}(q)$ contains lowercased, whitespace-delimited query tokens, ρ maps them to 4,096 hash buckets, and E contains 64-dimensional embeddings. The vector $g(q)$ has 42 numeric features. Proposal $\hat{r}$ selects among 15 policies. A delegated controller selects full history, cards, or windows for session access. The parameters $\theta = (\theta_1, \theta_2)$ include both stages' embeddings and classifiers. Appendix M gives layer dimensions and parameter counts.

A policy specifies a view, operator, evidence-unit budget, token budget, and formatter. Schema guards constrain the proposal using known task formats. The source adapter maps it to a supported action, consulting the delegated controller when configured. Write r_{exec} for this final policy, which can differ from $\hat{r}$. Its components $(s_{r_{\mathrm{exec}}}, A_{r_{\mathrm{exec}}}, k_{r_{\mathrm{exec}}}, L_{r_{\mathrm{exec}}}, \Phi_{r_{\mathrm{exec}}})$ define evidence assembly:

$$\begin{aligned} c(q) &= \Phi_{r_{\mathrm{exec}}}\big(q, A_{r_{\mathrm{exec}}}(q, \mathcal{V}_{s_{r_{\mathrm{exec}}}}, k_{r_{\mathrm{exec}}}); L_{r_{\mathrm{exec}}}\big), \\ p(q) &= [c(q); q], \qquad |\,\mathrm{tok}(c(q))| \le L_{r_{\mathrm{exec}}}. \end{aligned} \tag{4}$$

The formatter removes duplicate units, restores source order where required, retains source and speaker markers, and applies the reader-token cap. For LoCoMo, window access uses eight-turn windows at stride four and selects six windows before formatting. Evidence paths invoke the reader once, while direct structured paths return their result without generation. The full-context action is available as a policy, but the implementation does not estimate a calibrated probability that the selected evidence is sufficient.

Executed policies and source IDs provide an operational trace through guards, adapters, and operators. Appendix G gives concrete guard rules and delegated-action mappings.

Training and reuse. The first controller uses 26,725 route-labeled examples: 2,800 official training rows from BANKING77, CLINC150, TREC and HotpotQA-bridge, plus 23,925 procedural examples. The delegated controller uses 9,000 synthetic/procedural rows. Both minimize class-balanced cross entropy against prescribed access labels. The delegated labels follow template and length heuristics, independently of final benchmark answer scores. The same checkpoints serve all three readers, allowing backbone replacement without router retraining. This evaluation tests cross-reader portability within the represented task families. Appendix M specifies optimization, provenance, and splits.

Algorithm 1 Source preparation and query execution

Input: History $\mathcal{H}$, query q, frozen $\theta = (\theta_1, \theta_2)$, reader F
Output: Answer $\hat{Y}$

```
Prepare once: retain H and build supported views M
Obtain proposal r̂ using Equation 3
Apply matching schema guards to r̂
Map the guarded proposal to r_exec using the source adapter
    (consult θ_2 when configured and restrict to supported actions)
if r_exec specifies a direct resolver then
    return A_{r_exec}(q, V_{s_{r_exec}}, k_{r_exec})
else
    Assemble evidence c(q) using Equation 4
    return F([c(q); q])
end if
```

3.3 A Query through the Access Layer

LoCoMo history `conv-26` contains 419 turns in 19 sessions. The question *What did the charity race raise awareness for?* can be answered from a local exchange. From the same history, ARKA exposes raw session text, an extractive card, and overlapping dialogue windows. The session card is a compact search handle, while the window retains surrounding speaker turns. In the recorded execution, the delegated controller selects windows with probability 0.9843. Its six selected windows include `S2@0`, whose source turn `D2:2` contains *raising awareness for mental health*.

The Llama-3.1-8B reader answers *Mental health.* from 2,453 input tokens, versus 16,730 for the identical full-context answer, an 85.3% reduction. We use source annotations only for post-hoc analysis, not for retrieval. Appendix H provides the source identifiers and recorded predictions.

4 Theoretical Analysis

The preceding example retains a local exchange sufficient to answer one query. We now characterize when compact access preserves answer information. Let Y be a discrete answer with finite conditional entropy, Q the query, and W the reader's fixed side information. Let $C = f(\mathcal{H}, Q)$ include all evidence and metadata supplied by deterministic access. Write R^*_C and $R^*_{\mathcal{H}}$ for the minimum expected base-2 log loss given compact evidence and full history, respectively. Then

$$R^*_C - R^*_{\mathcal{H}} = H(Y \mid C, Q, W) - H(Y \mid \mathcal{H}, Q, W) = I(Y; \mathcal{H} \mid C, Q, W). \tag{5}$$

Here $H(\cdot \mid \cdot)$ is conditional entropy and $I(\cdot;\cdot \mid \cdot)$ is conditional mutual information, both in bits. The equality holds because C is determined by the source. Compact access is sufficient when the remaining history adds no answer information given C, Q, W, complementing capacity–error analyses of stored memory (Guo & Li, 2026).

Proposition 1 (relation-preserving access). Suppose an applicable operator returns a statistic $T(\mathcal{H}, Q)$ satisfying $Y \perp\!\!\!\perp \mathcal{H} \mid T, Q, W$ (conditional independence), and T can be recovered from (C, Q, W). Then $R^*_C = R^*_{\mathcal{H}}$. In particular, fix an unordered set S of m distinct values for one key. Let the entire history consist of these values in an order uniformly random conditional on S, Q, W, and let Q ask for the latest value. Any deterministic access representation invariant to permutation has exact-answer probability at most $1/m$ and Bayes log loss at least $\log_2 m$. An operator returning the last value answers exactly with one value.

Proof. Recoverability of T and its sufficiency give $H(Y \mid C, Q, W) = H(Y \mid \mathcal{H}, Q, W)$, proving the first statement by Equation 5. In the ordered-record example, each distinct value is last with probability $1/m$. An order-invariant message cannot distinguish those m answers. The last-value statistic retains exactly the required distinction. □

The proposition concerns relation preservation, which a retriever retaining update order can also provide. ARKA makes the required relation explicit: versioned access retains order, dialogue access

retains adjacency, and exemplar access retains pairing. Multi-record questions require broader evidence than a latest-value contract supplies. Appendix F gives the corresponding dense-information bound, motivating both narrow operators and broader evidence paths.

Sources of answer error. The sufficiency statement is conditional on choosing an applicable operator and retaining its support. To expose these two requirements, let V denote valid dispatch, and E denote that the selected evidence satisfies the chosen contract. For exact-answer error,

$$\Pr(\hat{Y} \neq Y) \leq \Pr(\neg V) + \Pr(V \cap \neg E) + \Pr(\hat{Y} \neq Y, V \cap E). \tag{6}$$

This decomposition separates operator applicability, evidence selection, and answering from valid evidence. Route-label accuracy measures agreement with dispatch supervision, while downstream scores assess the complete execution. Direct resolution bypasses generation when applicability and record selection are satisfied. For a finite reader, compact evidence can also reduce distraction: its expected log loss equals Bayes risk plus decoder mismatch (Appendix F). We use this framework to organize the analysis and evaluate the implementation through answer scores and service cost.

5 Experiments

We evaluate answer quality across backbones, examine structured execution and access selection, and measure construction and query costs under memory reuse and history growth.

5.1 Experimental Setup

We evaluate three readers—Llama-3.1-8B, Qwen3-8B and Qwen3.5-9B (Grattafiori et al., 2024; Yang et al., 2025; Qwen Team, 2026)—with a frozen controller. We evaluate 334 MAB questions across nine tasks, 1,986 LoCoMo conversational questions, 1,100 MemBench factual/reflective-memory questions, and 200 LongBench-5 examples each from QASPER, HotpotQA, MultiNews, TREC and passage retrieval (Hu et al., 2025; Maharana et al., 2024; Tan et al., 2025; Bai et al., 2024). MAB queries preassembled histories. The streaming study separately tests growth. Baselines retain their method-specific inference backends and internal call patterns. Appendix B specifies recoverable settings and timer boundaries.

Full context supplies the quality reference. LLMLingua-2-small uses an approximately 179M-parameter compressor, while A-Mem constructs reusable linked notes. Table 2 reports the BF16 core evaluation on A100-SXM4 80 GB GPUs and dual AMD EPYC 7763 hosts. Table 4 compares predictions from the LoCoMo component-cost runs, rescored with the same QA metric. Auxiliary aggregate comparisons appear separately in Appendix C.

MAB uses task-macro substring matching, LongBench-5 averages its task-specific metrics, and MemBench uses accuracy. LoCoMo combines token-overlap F1 on answerable questions with refusal accuracy on adversarial questions. All scores are on a 0–100 scale. A composite LoCoMo gain can therefore reflect answer recovery, refusal, or both. Appendix M gives task counts, decoding settings, metric details, and timing boundaries.

For a history serving Q_h queries, we distinguish reusable construction B_h from mean online access and reader costs:

$$\bar{T}_h(Q_h) = \frac{B_h}{Q_h} + \bar{t}_{\text{access}} + \bar{t}_{\text{reader}}. \tag{7}$$

Table 2 reports recorded cost ratios c_{full}/c_m on the A100 platform. For full context and ARKA, c_m measures reader generation or direct resolution. LLMLingua-2 additionally includes compression. Prompt assembly, tokenization, and CPU access are outside the generation timer. Llama-3.1-8B/Qwen3-8B ARKA timing uses a dedicated rerun with evaluation scores and baseline timings retained. Section 5.5 reports CPU access and construction separately to estimate service cost under Equation 7.

5.2 Main Results

Table 2 shows a favorable MAB operating point: ARKA improves full-context scores from 49.1 to 50.6, 47.5 to 53.9, and 53.6 to 55.8, with recorded cost ratios of 11.48, 8.55, and 2.56.

Table 2: Quality and recorded time ratios (Full / Method) on A100-SXM4 80 GB GPUs. Higher is better, and best values are bolded. LL2 denotes LLMLingua-2-small. Timing scope is defined in Section 5. CPU costs and auxiliary Qwen3.5-9B LoCoMo results appear in Section 5.5 and Appendix C.

		Score ↑			Recorded time ratio ↑	
Reader	Benchmark	Full	ARKA	LL2	ARKA	LL2
Llama-3.1-8B	MAB	49.1	**50.6**	31.4	**11.48**×	1.08×
	LongBench-5	**55.6**	53.7	49.2	**1.10**×	0.17×
	LoCoMo	38.3	**39.8**	30.1	**5.10**×	1.62×
	MemBench	**63.4**	62.5	58.9	**1.65**×	0.04×
Qwen3-8B	MAB	47.5	**53.9**	39.2	**8.55**×	0.81×
	LongBench-5	**59.9**	56.8	37.7	**1.02**×	0.24×
	LoCoMo	38.4	**39.2**	31.1	**4.19**×	1.72×
	MemBench	**68.9**	67.5	64.4	**1.77**×	0.07×
Qwen3.5-9B	MAB	53.6	**55.8**	41.6	**2.56**×	0.79×
	LongBench-5	**63.2**	52.1	41.2	**1.15**×	0.64×
	MemBench	**73.9**	69.1	67.0	**2.37**×	0.33×

Across all 11 comparable settings, ARKA exceeds LLMLingua-2 by 9.9 points on average, with a median 6.4× speedup in recorded processing time (Appendix B). LLMLingua-2 learns task-agnostic token retention (Pan et al., 2024), whereas ARKA selects an operator before assembling evidence. Proposition 1 motivates preserving update order and input–label bindings on structured tasks. Avoiding per-query compression also removes that component of access cost in Equation 7.

On LoCoMo, Llama-3.1-8B and Qwen3-8B achieve 39.8 and 39.2 versus full context's 38.3 and 38.4, with recorded query ratios of 5.10 and 4.19. The Llama-3.1-8B trace reduces mean reader input from 21,235 to 1,801 tokens, or 91.5%. The quality breakdown shows a trade-off between answer recovery and refusal: answerable-question F1 is 33.05 versus 36.64, while adversarial refusal accuracy is 63.00 versus 43.95. The paired history-bootstrap interval for the total-score difference is $[-0.98, 3.42]$ points and includes zero. Substantial input reduction is therefore the clearest measured benefit, alongside higher composite-score point estimates.

Performance varies with the task's access requirements. MemBench has recorded cost ratios of 1.65–2.37 with 0.9–4.8-point accuracy losses. On LongBench-5, recorded cost ratios are 1.02–1.15, with scores 1.9–11.1 points below full context (Table 2). These results place ARKA's strongest measured benefits in structured memory access. For broader document reasoning, Equation 5 motivates preserving jointly necessary evidence when assembling compact contexts. Improving this evidence coverage remains a direction for further development.

Comparison with retrieval and pruning methods. Table 3 compares RAPTOR (Sarthi et al., 2024) and Provence (Chirkova et al., 2025) in a separate Qwen3.5-9B suite. ARKA has higher reported scores on QASPER, MAB and LoCoMo, while Provence performs better on HotpotQA. ARKA has the highest recorded cost ratios on MAB and LoCoMo. RAPTOR retrieves original chunks and recursive summaries, while Provence jointly reranks and prunes query-relevant sentences. ARKA avoids summary-generation cost and explicitly preserves update order or dialogue adjacency, providing a mechanism consistent with its memory-task results and Proposition 1. Provence's HotpotQA advantage is consistent with learned sentence selection supporting distributed evidence, where sufficiency depends on joint coverage (Equation 5). Together, these patterns motivate matching access granularity to the source relations required by the query (Appendix C).

5.3 Structured Execution

On 214 aligned MAB questions across six tasks, ARKA scores 37.9 versus BM25's 27.9 at a 2,048-word cap and $k = 4$. On the single-record fact-update task, ARKA's latest-value resolver achieves 92.5% accuracy versus 17.5% for BM25, contributing $+12.5$ macro points while other tasks contribute -2.5.

Table 3: Retrieval and pruning comparison with Qwen3.5-9B. Recorded time ratios are Full / Method. Higher is better, and best values are bolded. This suite is separate from Table 2 and has aggregate-only records (Appendix C).

	Score ↑				Recorded time ratio ↑		
Dataset	Full	ARKA	RAPTOR	Provence	ARKA	RAPTOR	Provence
QASPER	48.7	**50.1**	43.7	46.4	0.99×	1.79×	**2.93**×
HotpotQA	**63.8**	50.0	50.1	55.8	2.55×	4.68×	**5.63**×
MAB	54.4	**56.4**	35.5	38.4	**4.81**×	1.96×	3.10×
LoCoMo	**41.7**	41.2	33.2	40.1	**6.83**×	5.89×	4.94×

Table 4: Comparison with A-Mem on LoCoMo using Llama-3.1-8B. Quality and warm-query statistics cover 1,986 questions from ten histories. Construction uses four histories with fresh-build records for both methods. *ARKA query time sums a separate CPU replay and reader timing, whereas A-Mem reports query wall time.

Metric	ARKA	A-Mem
Composite QA (1,986 questions)	39.78	43.43
Answerable F1 (1,540 questions)	33.05	33.48
Refusal accuracy (446 questions)	63.00	77.80
Mean build time (s/history)	0.057	3,414
Build time range (s/history)	0.031–0.114	2,800–4,069
Mean build LLM calls/history	0	2,022.5
Mean warm query time (s)	1.284*	1.245

The comparison evaluates the complete access layer, including 94 direct executions, against capped retrieval with different reader budgets. Latest-value resolution uses record order to distinguish the current answer from outdated matches (Proposition 1), avoiding generation when one record suffices. Appendix E reports task results, budgets, and execution counts.

5.4 Access-Selection Ablation

We compare schema enforcement on/off on 334 MAB questions with Llama-3.1-8B and Qwen3.5-9B in BF16 on RTX 3090. Query IDs, controller scores, operators and decoding remain fixed. These runs are separate from the A100 comparisons.

Task-macro gains are 3.81 and 1.67 points, with 95% task-resampling intervals $[0.00, 8.65]$ and $[0.00, 3.61]$ describing variation in task composition with checkpoints fixed. On BANKING77, Llama-3.1-8B gains 17.5 points while mean input falls from 4,504 to 4,376 tokens. Guards match all questions, change 82 policy labels, and alter 60 contexts or direct paths. The other 274 predictions are identical. This intervention measures schema enforcement within the learned-controller pipeline. The paired trace localizes the observed changes to evidence assembly or direct execution, illustrating how explicit access contracts make the source of an improvement inspectable. Appendix D gives all task scores and paired outcomes.

5.5 Efficiency and History Scaling

Table 4 compares ARKA and A-Mem using predictions from the timing runs, rescored with one evaluator on 1,986 questions from ten LoCoMo conversations. Composite QA is 39.78 for ARKA and 43.43 for A-Mem. Answerable F1 differs by 0.42 points before rounding, with refusal accuracy accounting for most of the composite gap.

For the four matched histories, ARKA's CPU builds average 0.057 s, while A-Mem's note construction averages 3,414 s and 2,022.5 LLM calls. ARKA constructs views for all ten histories in 0.464 s. Adding its replayed 0.121 s/query CPU preparation to 1.163 s reader time estimates 1.284 s warm service, compared with A-Mem's 1.245 s recorded mean.

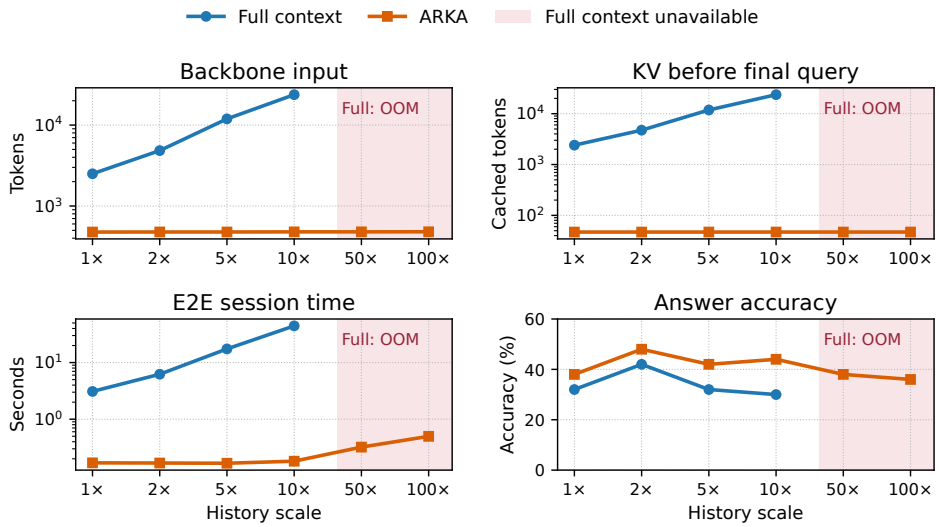


Figure 3: History growth with a fixed turn-pair path on 50 MemBench queries and unrelated prepended dialogues. Full context retains streamed history KV and exceeds A100 80 GB memory at 50×, while CPU access keeps about 480 input tokens through 100×. Shading marks scales where full context runs out of memory, with no accuracy or latency value assigned.

A-Mem uses LLM-generated descriptions, links, and memory evolution to enrich retrieval (Xu et al., 2025), while ARKA indexes source structure directly. This distinction trades construction cost against retrieval quality, with the observed quality gap concentrated in refusal accuracy. Equation 7 predicts that construction savings matter most for new histories and diminish per query with reuse (Appendix A).

CPU memory retains recoverable source text, metadata, and indices. Reader KV holds attention activations for a particular token sequence. Ordinary QA prefills a fresh compact prompt, reusing only identical prefixes when available. Thus persistent histories need not keep their full-history KV resident between requests.

Figure 3 tests this separation by expanding 50 fixed MemBench histories with other dialogues from the same subgroup. Full context already reuses streamed history KV: input grows from 2.5k to 23.8k tokens at 10×, and session time from 3.09 s to 44.48 s, before OOM at 50×. ARKA uses a fixed turn-pair path, retains the 47-token system-prefix KV before the final query, and assembles 477–480 reader-input tokens. At 100×, total session time is 0.498 s and accuracy is 36%, compared with 38% at 1×. With a fixed evidence budget, reader KV scales with the assembled context and generated answer rather than stored history. The nearly constant input lengths demonstrate how external storage decouples archive growth from active GPU state. This design suits assistants that retain many past sessions but answer queries from a small number of relevant exchanges.

6 Discussion and Conclusion

ARKA places typed access between persistent sources and generation. Its operators preserve update order, dialogue adjacency, and input–label correspondence. The analysis explains their information-preserving role, while experiments demonstrate inexpensive construction and compact reader inputs for structured and conversational memory. These costs favor services with many intermittently queried histories.

Dispatch combines a prescribed operator library with guards for known schemas. New operators require applicability rules and evidence-contract validation. The interface accommodates stronger retrieval for distributed evidence and broader reading when needed. Source pointers make access decisions and evidence recovery inspectable across backbones.

AI USE STATEMENT

Generative AI tools assisted with literature review, theoretical formulation and proof drafting, interpretation of existing experimental results, code and artifact checks, figure editing, and manuscript preparation. The authors are responsible for verifying the final claims, proofs, experimental evidence, citations, and artifacts.

REPRODUCIBILITY STATEMENT

Algorithm 1 and Appendix G specify the access pipeline and view contracts. Appendix F gives analytical assumptions and proofs. Appendices J and M specify training data and implementation details. Appendices A and B define scoring and timing protocols. The anonymous code supplement provides the ARKA implementation, two router checkpoints, router training data, and reproduction instructions.

REFERENCES

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.172/.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George B. M. van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, 2022. URL https://proceedings.mlr.press/v162/borgeaud22a.html.

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 335–336, 1998. doi: 10.1145/290941.291025.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pp. 38–45, 2020. doi: 10.18653/v1/2020.nlp4convai-1.5. URL https://aclanthology.org/2020.nlp4convai-1.5/.

Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. Provence: Efficient and robust context pruning for retrieval-augmented generation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5e956fef0946dc1e39760f94b78045fe-Abstract-Conference.html.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.17323v2.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

Anxin Guo and Jingwei Li. Hallucination is a consequence of space-optimality: A rate-distortion theorem for membership testing. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.00906v7.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-1902.

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*, 2020. URL https://proceedings.mlr.press/v119/guu20a.html.

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

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43, 2023. URL https://jmlr.org/papers/v24/23-0037.html.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, 2024. doi: 10.18653/v1/2024.naacl-long.389. URL https://aclanthology.org/2024.naacl-long.389/.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://aclanthology.org/2023.emnlp-main.825/.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://aclanthology.org/2024.acl-long.91/.

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*, 2020. URL https://aclanthology.org/2020.emnlp-main.550/.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 1311–1316, 2019. doi: 10.18653/v1/D19-1131. URL https://aclanthology.org/D19-1131/.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://proceedings.mlr.press/v235/lee24c.html.

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. In *Advances in Neural Information Processing Systems*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. URL https://aclanthology.org/C02-1150/.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6342–6353, 2023. doi: 10.18653/v1/2023.emnlp-main.391. URL https://aclanthology.org/2023.emnlp-main.391/.

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 (Volume 1: Long Papers)*, pp. 13851–13870, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560v1*, 2023. URL https://arxiv.org/abs/2310.08560v1.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024. URL https://aclanthology.org/2024.findings-acl.57/.

Qwen Team. Qwen3.5-9B model card, 2026. URL https://huggingface.co/Qwen/Qwen3.5-9B. Accessed 2026-09-17.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GN921JHCRw.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards more comprehensive evaluation on the memory of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19336–19352, 2025. doi: 10.18653/v1/2025.findings-acl.989. URL https://aclanthology.org/2025.findings-acl.989/.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mlJLVigNHp.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://aclanthology.org/D18-1259/.

Haozhen Zhang, Haodong Yue, Tao Feng, Quanyu Long, Jianzhu Bao, Bowen Jin, Weizhi Zhang, Xiao Li, Jiaxuan You, Chengwei Qin, and Wenya Wang. Learning query-aware budget-tier routing for runtime agent memory. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.06025v3.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://ojs.aaai.org/index.php/AAAI/article/view/29946.

A Quality Associated with Service-Cost Records

The service-cost comparison evaluates ARKA and A-Mem on the same 1,986 LoCoMo questions using Llama-3.1-8B. Each method's predictions are associated with its timing run and aligned to the canonical dataset by conversation identifier and query index. Question strings and categories are verified for every matched item.

Both sets of predictions are rescored with the same upstream LoCoMo QA function, using canonical dataset answers. Overall scores are 39.779179 for ARKA and 43.429939 for A-Mem. Answerable F1 scores are 33.052889 and 33.475233. Adversarial refusal accuracies are 63.004484 and 77.802691. Two stored A-Mem gold strings disagree with the dataset, both in the adversarial category where scoring ignores gold text and checks refusal phrases. Their metric effect is zero. No answerable-reference discrepancy is present. The source predictions are unchanged. Only scoring is unified.

The fresh-build comparison uses the four conversations for which A-Mem records new construction. ARKA's corresponding per-history CPU build times are selected from its ten-history replay:

Conversation	ARKA build (s)	A-Mem build (s)	A-Mem build calls
conv-41	0.114470	3760.533224	2171
conv-42	0.039936	3028.586348	2031
conv-44	0.040759	4068.549280	2224
conv-49	0.031489	2800.237110	1664
Mean	0.056664	3414.476490	2022.5

ARKA uses zero build-time LLM calls. A-Mem's other six histories use cached memory. The older ten-history cold-build estimate multiplies the four-build mean by ten. The main comparison reports the actual matched fresh-build subset. These descriptive construction statistics apply to the four observed histories, with extrapolated totals identified separately.

The raw ARKA generation mean is 1.162918 s and its separately replayed CPU preparation mean is 0.121022 s. Their sum estimates 1.283941 s warm service, excluding process startup and diagnostic generation probes. The associated A-Mem records have mean query wall time 1.245116 s. These are source-associated costs with explicit boundaries. ARKA's sum is not a joint wall-clock measurement. At Q queries per history, substituting the four-history construction means into Equation 7 gives

$$T_{\mathrm{ARKA}}(Q) = 0.0567/Q + 1.2839, \qquad T_{\text{A-Mem}}(Q) = 3414.4765/Q + 1.2451 \quad \text{seconds.} \tag{8}$$

These projections hold the measured query means fixed and vary amortization of the matched construction means, rather than measuring new query workloads. A separate trace measures 1.44 ms for the two neural controllers, excluding retrieval and assembly. Appendix I details the controller timings and their use in serial-cost projections.

B Measurement Boundaries

The core tables retain recorded benchmark scores and timing fields. For local full-context and ARKA reader paths, the query timer generally wraps model generation after prompt assembly, chat formatting, tokenization, and device transfer. A separate one-token prefill probe is excluded. Direct ARKA paths use a resolver execution timer. LLMLingua-2's recorded cost adds its per-query compression pass. A-Mem's query field can contain retrieval and multiple internal model calls, while its cost field adds amortized reusable construction. These fields do not share a complete wall-clock boundary.

The abstract summarizes the 11 comparable BF16 benchmark–backbone settings in Table 2, weighting each setting equally. The mean ARKA-minus-LLMLingua-2 score difference rounds to 9.9 points. For speed, we divide LLMLingua-2's recorded compression-plus-generation time by ARKA's recorded query time within each setting, then take the median. Using the underlying timing values gives 6.432801×, reported as 6.4×. Both abstract comparisons use LLMLingua-2 as the reference, whereas the table's individual cost columns use full context. The summary retains the timing variants represented in the table, including the original MemBench times for Llama-3.1-8B and Qwen3-8B. CPU preparation is characterized separately by the LoCoMo replay and is not added

Table 5: Reusable agent memories. Q and C denote recorded query and cost ratios relative to full context. A-Mem includes its reported reusable build charge in C. Method-specific timer boundaries differ. Complete CPU-plus-reader costs appear in Section 5.5.

		Full-context			ARKA			A-Mem		
Backbone	Benchmark	Score	Q	C	Score	Q	C	Score	Q	C
Llama-3.1-8B	MAB	49.1	1.00×	1.00×	**50.6**	**11.48**×	**11.48**×	47.5	6.73×	1.41×
Llama-3.1-8B	LoCoMo	38.3	1.00×	1.00×	39.8	**5.10**×	**5.10**×	**44.4**	2.90×	0.19×
Qwen3-8B	MAB	47.5	1.00×	1.00×	**53.9**	**8.55**×	**8.55**×	48.4	3.43×	0.32×
Qwen3-8B	LoCoMo	38.4	1.00×	1.00×	39.2	**4.19**×	**4.19**×	**41.9**	2.86×	0.30×
Qwen3.5-9B	MAB	53.6	1.00×	1.00×	**55.8**	**2.56**×	**2.56**×	54.2	1.83×	0.25×

to settings without corresponding measurements. Thus this summary describes recorded processing time, with full service accounting addressed by the component analysis.

Llama-3.1-8B/Qwen3-8B ARKA query times come from a dedicated controlled-timing pass over the frozen configuration. The aggregate preserves scores, input lengths, and baseline times from the quality evaluation. For example, Llama-3.1-8B MAB ARKA query time changes from 0.997376 s to 0.576344 s while the full-context reference remains 6.613515 s. Llama-3.1-8B LoCoMo changes from 1.162919 s to 0.633800 s against 3.231613 s. Their displayed ratios are therefore separately measured timer ratios, not contemporaneous full-service throughput. ARKA aggregate cells without separately recorded construction charges use the query field for the cost field as well.

Core local MAB generation is capped at 50 tokens with a 32,000-token context limit. LongBench caps are 128 for QASPER, 32 for HotpotQA, 512 for MultiNews, 64 for TREC, and 32 for passage retrieval. LoCoMo uses a 64-token generation default and a 131,000-token input limit, while MemBench uses an eight-token multiple-choice cap. These are recovered core-runner settings. A-Mem uses its own inference workflow: one recovered Qwen3.5-9B MAB TREC run uses vLLM, an internal 256-token cap, and two query LLM calls on average. Consequently, a shared reader family does not imply identical internal generation budgets or inference backends across methods.

The component record in Section 5.5 measures ARKA construction and CPU preparation through a replay without reader generation, then adds the associated evaluation's reader-generation mean. The replay matches 1,695 window and 291 card actions. A-Mem's cold-build total extrapolates the mean of four fresh histories to all ten. The other six query memories were cached. The 1.284 s ARKA online and 14.36× build-aware figures are additive estimates, not jointly measured wall times. Build-aware cost divides construction by 1,986 questions and adds query-weighted online cost. Equal weighting of histories with unequal query counts would define a different aggregate.

C Auxiliary Aggregate Comparisons

Table 5 retains the broader agent-memory aggregates. Its method-specific timing fields differ from the associated-run comparison in Table 4. Scores or timings from the two sources are not interchanged.

Table 3 reports an additional Qwen3.5-9B comparison at the aggregate level. The source experiment table supplies scores and timing ratios. Per-query records and a complete RAPTOR/Provence configuration manifest were not recovered, so interpretation is restricted to these within-suite aggregates. Its MAB reference scores differ from Table 2, with the configuration difference unresolved in the retained records. We therefore report the suites separately. Timing ratios follow the source table's build-aware accounting convention, with complete wall-clock measurement unverified.

A Qwen3.5-9B ARKA LoCoMo run has runtime proxy score 40.6 and is not commensurate with QA-scored baselines. A complete RTX 3090 run on 1,986 questions gives QA score 40.9003 and mean reader input 1,847.629 tokens. These measurements are not combined with the A100 timing ratios. The main tables restrict comparative cells to a shared quality metric.

D PAIRED ACCESS-SELECTION ANALYSIS

Each reader evaluates 334 questions, comprising eight 40-question tasks and 14 DetectiveQA questions. The guard-off and guard-on configurations share sample IDs, raw router scores, decoding settings and action implementation. The intervention changes only whether schema rules constrain the proposal. This paired design estimates the effect of schema enforcement on a fixed familiar-schema test set, holding controller training and task schemas constant.

Table 6: Paired access-selection results on 334 MAB questions. Off/on denotes schema enforcement with the learned controller fixed. Both exemplar conditions invoke the reader.

		Llama-3.1-8B		Qwen3.5-9B	
Task	n	Off	On	Off	On
BANKING77	40	65.0	82.5	90.0	95.0
CLINC150	40	72.5	75.0	85.0	92.5
DetectiveQA	14	35.71	50.0	50.0	50.0
Event QA	40	57.5	57.5	70.0	72.5
Single-record facts	40	92.5	92.5	92.5	92.5
Multi-record facts	40	2.5	2.5	2.5	2.5
LongMem	40	12.5	12.5	7.5	7.5
RULER	40	12.5	12.5	20.0	20.0
TREC	40	60.0	60.0	75.0	75.0
Task macro	334	45.63	49.44	54.72	56.39
Question mean	334	46.41	49.40	55.09	56.89

Schema rules match all 334 questions, change 82 route labels, and alter 60 executed contexts/direct paths. On the 274 unchanged executions, predictions are identical. Fourteen DetectiveQA queries move from reader generation to direct option resolution, reducing total reader calls from 254 to 240. The two fact tasks already use direct resolution in both conditions. BANKING77 and CLINC150 retain reader calls. Their intervention changes exemplar policies and evidence assembly.

For the question-mean difference, a paired percentile bootstrap over 12 task/context clusters (10,000 replicates, seed 20260915) gives Llama-3.1-8B $+2.99$ points with 95% interval $[0.29, 7.40]$, and Qwen3.5-9B $+1.80$ with $[0.00, 3.95]$. A separate bootstrap resampling the nine task-level differences (10,000 replicates, seed 20260916) gives task-macro intervals $[0.00, 8.65]$ and $[0.00, 3.61]$. These latter intervals characterize variation under resampled task composition with model checkpoints fixed. Each estimate retains its corresponding aggregation unit, while variability across retraining or new schemas requires a separate evaluation.

E STRUCTURED-ACCESS DIAGNOSTIC

The six-task MAB diagnostic comprises 40 questions each for single-record facts, multi-record facts, event QA, LongMem and RULER, plus 14 detective questions. All methods use aligned questions and answers. Scores below are task-macro means of the stored substring-match scores. We compare BM25 with maximal marginal relevance (MMR) reranking (Carbonell & Goldstein, 1998). The capped retrieval runner uses four ranked chunks and truncates in rank order using regex word tokens, not reader tokens. The 1,024 and 2,048 caps were evaluated in the same sweep. The stored records do not establish an independent selection rule for choosing one operating point. Both capped and uncapped points are therefore reported.

Method	Score	Recorded input	Query time (s)
Full context	34.11	23,541	6.494
BM25, 1,024-word cap	22.92	1,636	0.998
BM25+MMR, 1,024-word cap	22.92	1,636	1.407
BM25, 2,048-word cap	27.92	3,041	1.183
BM25+MMR, 2,048-word cap	27.92	3,041	0.909
BM25, uncapped	38.63	13,460	1.726
BM25+MMR, uncapped	38.21	13,460	1.703
ARKA, mixed execution	37.92	mixed units	0.750

The original ARKA input field is not a reader-token measure on direct paths: it records output-word counts for the 94 directly resolved questions. The remaining 120 questions invoke the reader. We consequently do not interpret the historical 2.8k mixed average as a matched input budget. BM25 and BM25+MMR at the 2,048 cap retain identical chunk IDs, lengths and predictions for all 214 questions. Their timing difference is a difference between runs, not evidence that MMR changes selected information at this cap.

Task	Questions	BM25 score	ARKA score	ARKA reader calls
Single-record facts	40	17.5	92.5	0
Multi-record facts	40	0.0	2.5	0
Event QA	40	62.5	57.5	40
Detective	14	50.0	50.0	0
LongMem	40	15.0	12.5	40
RULER	40	22.5	12.5	40

The task breakdown distinguishes two access requirements. Latest-value resolution matches the single-record task, whereas multi-record questions require composition across facts beyond this operator's contract. Reporting direct-call coverage together with per-task accuracy separates computational savings from answer correctness, following the applicability and evidence conditions in Equation 6.

F Information-Access Analysis

Analytical setting. The source history is $\mathcal{H}$, and W represents fixed reader-side information, such as pretrained knowledge and task instructions, shared by the compact-context and full-history comparisons. The message C includes all source-derived text, metadata, and route identifiers visible to the reader. The risks R^*_C and $R^*_{\mathcal{H}}$ are Bayes log-loss risks over conditional answer distributions. They describe information availability separately from the measured performance of a particular reader. The sufficiency assumption $Y \perp\!\!\!\perp \mathcal{H} \mid T, Q, W$ states that the operator's statistic T retains all answer information supplied by the history conditional on Q, W.

We analyze information preservation under the stated access assumptions. All entropies and losses use base-2 logarithms. The results characterize optimal probabilistic decoding. Trained-router performance and empirical F1 are evaluated separately in the experiments. The membership-testing analysis of Guo and Li motivates the capacity question, while the derivations below concern query-conditioned access.

Information left behind by access. Let $(\mathcal{H}, Q, Y, W)$ have a joint distribution with finite answer entropy, and let $C = f(\mathcal{H}, Q)$ be the complete deterministic message visible to the decoder. Include route IDs or other side channels in C if the decoder receives them. Define Bayes log-loss risks $R^*_C = \inf_p \mathbb{E}[-\log_2 p(Y \mid C, Q, W)]$ and $R^*_{\mathcal{H}} = \inf_p \mathbb{E}[-\log_2 p(Y \mid \mathcal{H}, Q, W)]$ over all conditional distributions. Then $R^*_C - R^*_{\mathcal{H}} = I(Y; \mathcal{H} \mid C, Q, W)$. For a stochastic access rule, the same statement holds after conditioning on its independent random seed.

Proof. The log-loss Bayes risk equals the relevant conditional entropy, because expected cross entropy equals entropy plus a nonnegative conditional KL divergence. Since C is determined by $(\mathcal{H}, Q)$, $H(Y \mid \mathcal{H}, C, Q, W) = H(Y \mid \mathcal{H}, Q, W)$. Subtracting this from $H(Y \mid C, Q, W)$ gives the claimed conditional mutual information. Equality to zero is equivalent to conditional independence

of Y and $\mathcal{H}$ given (C, Q, W). It can hold for a short sufficient message even when the archive is large. □

Corollary: dense answer information. Suppose a query asks for an r-bit vector Y that is uniformly distributed conditional on Q, W, and perfectly determined by the source. Suppose the complete transmitted message has at most 2^b possible values for each Q, W (e.g., a fixed b-bit encoding, including metadata and length). Then $H(Y \mid C, Q, W) \geq r - b$ for $b < r$. Indeed, $H(Y \mid Q, W) = r$ and $I(Y; C \mid Q, W) \leq H(C \mid Q, W) \leq b$. Moreover, exact vector recovery succeeds with probability at most 2^{b-r}: there are 2^r equiprobable answers, while one deterministic decoder can correctly name at most one answer for each of at most 2^b messages. Randomizing the decoder cannot improve this maximum.

This explains a possible cost of a fixed evidence budget, not a universal penalty for long histories. For example, if N independent facts are stored and Q names just one, an exact query-conditioned lookup needs only that fact's value (given a known address). If one query instead requests all N facts, the short-message guarantee fails. Preserving the archive prevents irreversible storage loss but does not remove the communication requirement at answer time. For a K-symbol vocabulary and exactly L transmitted tokens, a counting upper bound is $L \log_2 K$ bits. Natural-language tokens are neither independent nor a measured answer-information capacity.

Finite readers and observed scores. For a fixed decoder p_ϕ, its expected log loss given C decomposes exactly as

$$\begin{aligned}\mathbb{E}[-\log_2 p_\phi(Y \mid C, Q, W)] = H(Y \mid \mathcal{H}, Q, W) + I(Y; \mathcal{H} \mid C, Q, W) \\ + \mathbb{E}\,\mathrm{KL}\big(p^*(Y \mid C, Q, W) \| p_\phi(Y \mid C, Q, W)\big).\end{aligned}$$

The first term is residual ambiguity under full source access, the second is access-induced uncertainty, and the third is decoder mismatch. A shorter context can increase the second term but reduce the third by removing distracting input. The decomposition therefore describes how compact access can improve a finite reader's predictions while preserving the information inequality. These terms provide an analytical framework, with token-F1 reported as an empirical answer metric. Measuring evidence coverage separately from answer generation would connect the framework to document-reasoning outcomes.

Variability and benchmark uncertainty. Let X include a sampled history, query and annotation, Θ denote a router training realization, Ξ denote reader sampling/runtime randomness, and $S = S(X, \Theta, \Xi)$ be the scored answer. Repeated total variance gives

$$\mathrm{Var}(S) = \mathrm{Var}_X \mathbb{E}[S \mid X] + \mathbb{E}_X \mathrm{Var}_\Theta \mathbb{E}[S \mid X, \Theta] + \mathbb{E}_{X,\Theta} \mathrm{Var}(S \mid X, \Theta).$$

These terms separate item variation, training variation, and conditional execution variation. The evaluation fixes the router and uses greedy decoding, while variation across retraining and hardware requires repeated runs. Questions sharing a history are dependent, motivating the history-level resampling used in the paired analysis. Annotation ambiguity and refusal rules are additional sources of scoring uncertainty whose rates remain unmeasured. For reference, 50 independent binary trials have maximal standard error $1/(2\sqrt{50}) = 7.07$ percentage points. This generic calculation illustrates the precision associated with that sample size, rather than providing a confidence interval for the paired growth-study examples. The growth study therefore emphasizes capacity and input scaling.

G View Contracts, Robustness, and Extension

Symbols in the main method have the following roles: $\mathcal{H}$ is the ordered source archive, $\mathcal{S}$ indexes legal view types, B_s builds view $\mathcal{V}_s$, z_j is a unit's retrieval text, τ_j is its time/position, m_j contains source pointers and metadata, ψ scores source sentences, $h(q)$ is the router input, $\hat{r}$ is the router proposal, r_{exec} is the final executed policy, and (s, A, k, L, Φ) denotes a policy's view, operator, unit budget, token budget, and formatter. The context $c(q)$ is assembled evidence and $p(q)$ is the final reader prompt. A view is legal only if the source adapter can supply its required structure.

View	Construction/index unit	Query and assembly contract
Card/chunk	Deterministic source splitting, salience-selected sentences, raw-source IDs and sentence offsets	Score card access text, rank recoverable sentences, expand by configured radius, de-duplicate and recover original spans.
Dialogue	Speaker-tagged turns grouped within sessions	Score full windows. The LoCoMo configuration uses eight turns, stride four, top six windows, then restores session/start order and speaker markers.
Turn pair	Adjacent exchanges from a dialogue	Preserve the local question/response relation when selecting recent or relevant exchanges.
Fact	Explicit records with serial/update order	Lexically shortlist candidates, select the latest eligible record. This operator assumes the query can be answered by a single appropriate record.
Exemplar	Supplied input–label pairs	Rank example inputs and restore their labels, assemble flat or label-grouped demonstrations.
Fallback	Source text in its original format	Full reading within the reader limit. This is a selectable action, not a calibrated confidence test.

For card recovery, the retained core typically has two to four sentences. Query-time lexical sentence scores and overlap select spans with a core-sentence bonus. A configured neighboring-sentence radius restores local context. Multiple-choice evidence operators score question–option support and may return the selected option directly. Assembly outputs source text or a structured answer. It does not ask an LLM to synthesize the memory index. The release contains both payload-level routes and benchmark adapters: the first-level route name therefore does not alone specify which final operator ran. The route-to-family mapping in Appendix L is a taxonomy, not an executed-action count.

The multiple-choice resolver ranks question–option probes using BM25 plus lexical-support bonuses. It uses three chunks for general questions, five sentences for culprit questions, or eight sentences for identity questions, then returns the highest-scoring option without generation. This lexical score is not a calibrated correctness probability.

For example, the classification guard recognizes *provided mapping* and *numerical label*, selecting flat assembly of 24 exemplars. Domain cues such as *calendar* select label-grouped assembly with a 24-example budget, at most six labels and six examples per label. A conversation adapter can delegate window/card selection independently of the proposal's family.

When distractor rejection is justified. For a ranking operator, let the smallest score of the selected top-k set exceed the largest excluded score by $\gamma > 0$. If a perturbation changes every candidate score by less than $\gamma/2$, the selected set is unchanged: the worst selected-minus-excluded margin remains positive. Likewise, an argmax route remains unchanged if its winning logit margin exceeds twice a uniform logit-perturbation bound. These are sufficient stability conditions. Training does not certify them. A same-entity distractor can violate the retrieval margin, and a synonym can alter lexical overlap even if the router chooses the intended operator. Training includes natural-style query templates, and structured version/adjacency rules supply useful inductive biases, but the unrelated-history stress test does not measure these harder perturbations.

For latest-value lookup, rejection of old versions is guaranteed only when the shortlist contains the correct entity/relation records, their serial numbers encode the intended update order, and the answer is determined by the last record. Incorrect entity matching, contradictory sources, and multi-record reasoning violate that contract. Thus robustness should be tested at both dispatch and evidence-selection stages rather than inferred solely from a correct route label.

Cost of extending the action library. The 15-way first-level head has 128 input features per output. Adding one route while keeping the encoder/hidden layers fixed adds 129 parameters (516 bytes in FP32) and 128 multiply-accumulates per query in the output layer. A new route that reuses an existing view needs no new archive copy. A genuinely new view needs a deterministic builder, index, applicability rule, and formatter. Its storage is $O(N)$ times its per-unit representation and

its build cost depends on the operator. Extension therefore combines a small classifier update with operator-specific construction and validation.

The released training logs record 80.961 s for the first router and 8.0555 s for the delegated router. The first uses 21,380 fitting rows, 5,345 held-out rows, 80 epochs and batch size 64, or 26,800 fitting updates. An illustrative additional 1,000 fitting rows at the same settings gives 28,000 updates, 4.48% more. Linear scaling of that training log suggests about 3.6 extra seconds. This is an arithmetic projection, not a measured retraining experiment, and excludes label design and validation. Automatically discovering view types is a plausible extension of the small controller, but the present release learns dispatch over a prescribed action library.

H Conversation-Memory Case Study

The following LoCoMo example traces the stages illustrated schematically in Figure 2 using a recorded execution. History `conv-26` contains 19 sessions and 419 turns. Query index 82 asks: *What did the charity race raise awareness for?* The gold answer is *mental health*.

From this one history, raw session `S2` retains its original dated speaker turns. Its extractive card stores core sentence indices 2, 3, 11, and 16 of 53. Dialogue window `S2@0` contains the first eight turns. These are three views of the same source. The card's first retained sentence is a date, illustrating that it is an access key rather than an LLM-written semantic summary. Sentence-list indices differ from the benchmark's dialogue-turn identifiers. Fact/update and labeled-example views are not manufactured from this history without the corresponding input schema.

The saved first-level route is local dialogue continuity. In this adapter, the active delegated router selects window access with probability 0.9843, against 0.0157 for card and 0.000036 for full. The selected windows are `S2@0`, `S3@0`, `S7@0`, `S8@12`, `S8@16`, and `S14@24`, in source order. Window `S2@0` contains Caroline's turn `D2:2` with the phrase *raising awareness for mental health*. Selected window/session IDs match the archived trace. The formatted windows preserve speaker and turn markers before insertion into the benchmark answer prompt.

ARKA outputs "Mental health" using 2,453 input tokens. Full context produces the same answer using 16,730 input tokens. The input reduction is $1 - 2453/16730 = 85.34\%$. This selected case illustrates evidence retention with a smaller prefill, complementing the aggregate evaluation. The source annotation locates support after evaluation and was not passed to retrieval. CPU reconstruction reproduces the selected units. Verification is at the source-selection level because historical final-prompt bytes were not stored.

I Timing, KV Cache, and Evaluation Metrics

Equation 7 defines build-aware service cost. Appendix A gives the measured components and amortization, and Appendix B specifies timer boundaries and the separate diagnostic prefill probe. Construction is charged once per memory for both ARKA and A-Mem. Query-dependent compression belongs to online access cost unless the implementation reuses its output.

The generic QA runner tokenizes a complete prompt and calls generation anew, without passing prior-question KV. This differs from reusing the CPU archive, which remains available across questions.

The streaming experiment uses a separate session implementation: full context commits each history turn to the causal cache, while ARKA retains the history externally and reuses the fixed system-prefix state. The cache for a token depends on its preceding tokens and positions. Once selected evidence is reordered or replaced, only an identical prefix can generally be reused. CPU source pointers are not reusable backbone KV. The preserved 47-token pre-query cache therefore describes that system prefix, not a compressed cache encoding the full dialogue. The stress path uses configured turn-pair access rather than the learned two-level controller.

The later Qwen3.5-9B LoCoMo run records 0.595 ms and 0.843 ms mean router times. Their sum gives $100(0.000595 + 0.000843) = 0.1438$ CPU-seconds of serial routing work per second at 100 QPS, under a linear no-contention projection. This calculation estimates controller demand from measured serial timings. Retrieval, view construction, Python scheduling, tokenization, and

memory bandwidth contribute additional service costs. Evaluating throughput and tail latency under saturation requires concurrent measurements of the complete pipeline.

I.1 Metric definitions and auxiliary runs

On LoCoMo, answerable questions use normalized, stemmed token F1. The multi-hop category matches comma-separated answer items and the open-domain category uses the text preceding a semicolon. Adversarial queries score one when the lowercased prediction contains either *no information available* or *not mentioned*, and zero otherwise. The overall QA score averages 1,540 answerable and 446 adversarial items and multiplies by 100. It is not answerable-question F1 alone. A Qwen3.5-9B ARKA run has runtime proxy score 40.6, which is excluded from comparative QA cells because it is not commensurate with full-context QA 41.0. A separate RTX 3090 run scores 40.9003 under the QA definition, with 1,847.629 input tokens. Its quality and timing are not combined with A100 runs. Table 8 reports proxy LoCoMo scores for precision transfer and labels that metric explicitly.

J Training Provenance and Scope

The first-level training corpus contains 800 BANKING77 training rows (Casanueva et al., 2020), 800 CLINC150 training rows (Larson et al., 2019), 800 TREC training rows (Li & Roth, 2002), and 400 HotpotQA-bridge training rows (Yang et al., 2018). The remaining 23,925 rows are deterministic schema/procedural examples, including natural-style episodic/dialogue templates and explicit prompt forms. Names, topics, and instruction skeletons are drawn from construction code, not saved evaluation answers. The 800 CLINC150 development rows were excluded before final training. The fitting/held-out partition is within the training corpus, and checkpoint selection uses this held-out portion rather than final benchmark scores. The supplementary code's training-data summaries and provenance notes record the sample sources and exclusion rules.

The delegated router uses 9,000 synthetic/procedural examples. Query templates and predeclared length heuristics supply its action labels. The generation pipeline excludes final benchmark IDs, answers and model predictions. A reproducible exact-string audit finds no overlap between the 4,106 distinct normalized first-level training questions and 1,974 distinct LoCoMo questions. This verifies exact-string separation for the audited questions. HotpotQA training and evaluation share a task family, and known schema phrases can recur. Semantic and unseen-schema transfer require grouped or held-out-schema evaluations beyond this within-corpus split.

The evaluated policy library spans five broad families and 15 implemented policies for document, conversation, update, event, exemplar, and format-sensitive workloads. The controller is reused across readers without retraining. Extending to new relation structures may involve a new operator, training examples, and validation. The small neural dispatcher limits the parameter cost of this extension, while applicability remains defined by each operator's source and evidence requirements.

K Limitations

ARKA's quality–cost trade-off depends on how readily the query's evidence can be localized. Global-reading tasks and format-sensitive inputs can benefit from broader or full-context access, as the LongBench-5 results illustrate. The full-context route is selected through the current dispatch mechanism, without a calibrated evidence-sufficiency check. CPU construction and indexing are reusable, making their amortization particularly relevant to histories queried repeatedly. The lightweight controller can be retuned for deployment-specific workloads while remaining separate from the backbone. Further development includes evidence-sufficiency checks, transfer to new schemas, and production policies for deletion, privacy, conflicting instructions, and auditability.

The evaluation measures the complete access layer and the schema-guard intervention. Isolating the independent gain from learned dispatch requires a comparison against rules over the same operator library. The history-growth experiment uses a fixed turn-pair policy with unrelated added dialogues. Paraphrases and same-entity distractors require complementary tests of dispatch and retrieval robustness.

L ADDITIONAL METHOD DETAILS

Table 7 groups the first-level router's 15 canonical routes into five families. These groups organize the policy library, while source adapters determine the final executed action.

Table 7: Route families. The released first-level router predicts 15 canonical routes, which are grouped here into five conceptual memory-access families.

Route family	Canonical routes
Fact/event	sparse event recall, multi-fact lookup, mutable fact stream, needle lookup, option-evidence probe
Exemplar	flat exemplar pairs, label-grouped exemplar pairs
Dialogue/session	episodic conversation memory, local dialogue continuity, session archive memory
Document	bridge-document QA, long-document lookup, multi-source coverage summary, format-sensitive document QA
Fallback	conservative full-context fallback

For textual retrieval inside a selected view, ARKA uses BM25 (Robertson & Zaragoza, 2009). For a view with N_s units and query-token sequence $\mathrm{tok}(q)$,

$$\begin{aligned} \mathrm{BM25}(q,u) &= \sum_{a \in \mathrm{tok}(q)} \mathrm{idf}(a) \frac{\mathrm{tf}(a,u)(k_1+1)}{\mathrm{tf}(a,u) + k_1\left(1 - b + b|u|/\overline{|u|}\right)}, \\ \mathrm{idf}(a) &= \log\left(1 + \frac{N_s - \mathrm{df}(a) + 0.5}{\mathrm{df}(a) + 0.5}\right). \end{aligned} \tag{9}$$

Here $\mathrm{tf}(a,u)$ is the count of token a in unit u, $\mathrm{df}(a)$ counts units containing a, and $|u|$ and $\overline{|u|}$ are unit length and mean unit length in retrieval tokens. The implementation uses $k_1 = 1.5$ for term-frequency saturation and $b = 0.75$ for length normalization. Repeated query tokens contribute repeatedly to the sum. Structured paths can replace this ranker with a resolver, such as latest-value lookup. View contracts are specified in Appendix G, and training provenance in Appendix J.

For the CPU salience score ψ used in document-card construction, the released builder scores each candidate sentence using lightweight features: moderate sentence length receives a positive weight, numeric tokens and entity-like capitalized tokens receive bonuses, and very short fragments, unbalanced quotation fragments, reporting-only utterances, or connective prefixes receive penalties. Candidate sentence indices are filtered to the valid source range, de-duplicated while preserving order, and truncated to the configured limit. Salience refinement retains two to four core sentences when available. Query-time span recovery can then re-rank sentences inside selected cards using lexical match to the query, token overlap, and a small bonus for sentences already marked as core, before expanding each selected sentence by a local source-sentence radius.

M REPRODUCIBILITY DETAILS

This appendix records the implementation details, data provenance, and evaluation protocol behind the reported comparisons. It clarifies which parts of ARKA are trained, cached, or counted in timing.

M.1 RELEASED ARTIFACTS AND CLEAN TRAINING DATA

The anonymous artifact contains the CPU routers, memory builders, benchmark runners, router supervision files, and reproduction notes. The summary below lists the components used by the experiments.

Released artifact summary.

Artifact	Contents and role
First-level router checkpoint	Fifteen-route CPU workload router selecting the canonical ARKA memory-access policy.
Delegated memory-policy checkpoint	Three-action CPU router used only inside episodic/session memory routes, selecting full, card-summary, or dialogue-window access.
Router supervision data	Clean route-labeled JSONL files for the two routers, with explicit provenance summaries and no final-evaluation questions, answers, predictions, or sample ids.
Memory construction code	CPU builders for cards, chunks, dialogue windows, turn pairs, fact records, exemplar banks, lexical indices, and prompt assembly.
Evaluation runners	Benchmark-specific runners that share the same backbone model, prompt surface, decoding settings, and scoring script within each benchmark cell.
Reproduction notes	Training-data provenance, environment notes, and commands for CPU checks, router training, and benchmark evaluation.

Appendix J specifies training sources, generated supervision, exclusions, and the fitting/held-out split. Final benchmark answers and predictions are excluded from controller supervision.

M.2 ROUTER ARCHITECTURE AND OPTIMIZATION

The first-level router follows Equation 3. Its token sequence $\mathcal{G}(q)$ is lowercased and split on whitespace, with no explicit n-gram expansion. The hash ρ uses an eight-byte BLAKE2b digest modulo 4,096. The 64-dimensional embedding table E is averaged by an `EmbeddingBag` and concatenated with the 42-dimensional numeric vector $g(q)$ to form $h(q)$. An empty token sequence uses bucket zero. The classifier has two 128-unit ReLU hidden layers and a 15-way output head. The full first-stage parameter set θ_1 contains 294,287 parameters, including E.

The delegated memory-policy router has the same lexical hashing design, but it also embeds the memory schema. Its input concatenates a 64-dimensional token hash vector, an 8-dimensional schema embedding, and 32 numeric features. The MLP has two 96-unit ReLU hidden layers and a 3-way action head. This checkpoint contains 281,843 learned parameters. Together, the released ARKA routers contain 576,130 learned parameters, matching the 0.58M parameter count reported in the main text.

Both routers are optimized with AdamW on CPU-compatible PyTorch modules. The first-level router uses learning rate 2×10^{-3}, weight decay 10^{-4}, batch size 64, 80 epochs, class-balanced cross entropy, and selects the checkpoint by held-out macro-F1 within the clean training JSONL. The delegated memory-policy router uses learning rate 2×10^{-3}, weight decay 10^{-4}, batch size 256, and 40 epochs. Training is lightweight, with the released runs recording about 81 seconds for the first-level router and about 8 seconds for the delegated memory-policy router on the same CPU node. At inference time, both routers run on the host CPU and never call an LLM.

M.3 Memory Construction and Assembly

Appendix G specifies the supported views and their assembly contracts. The final policy r_{exec} in Algorithm 1 selects the view, operator, budgets, and formatter. Appendix L specifies lexical scoring and sentence selection.

The core local full-context, ARKA, and LLMLingua-2 comparisons use the same answer template after context selection:

```
System instruction.
Context or working context selected by the method.
Question.
Answer generation prompt.
```

Full-context prompting uses the uncompressed input history or document as the context. ARKA uses the route-conditioned working context. LLMLingua-2 receives the same gathered context before compression and then uses the compressed text in the same answer template. The auxiliary RAPTOR/Provence suite has only aggregate provenance, as specified in Appendix C. Core chat-model runs that expose a thinking mode use the no-thinking template.

M.4 Timing and Evaluation Protocol

Recorded query and cost ratios use the method-specific fields defined in Appendix B. Local generation-only timers exclude host assembly and tokenization, whereas the component trace separately measures CPU preparation. Cost fields add preprocessing only where that charge is recorded. The core aggregate ratios must therefore be distinguished from the explicitly instrumented CPU-plus-reader service analysis.

The core local-reader comparisons use the designated backbone and precision. Methods with internal model calls retain their own backend and generation settings (Appendix B). Decoding is deterministic (`do_sample=false`) in the local transformer backend. Benchmark-specific maximum generation lengths follow the corresponding runner. Scores are aggregated over the same evaluated examples, and sharding only changes wall-clock completion time, not per-example timing. Reported GPU timings use one NVIDIA A100-SXM4 80 GB GPU per backbone worker. CPU-side ARKA construction and routing run on the host dual AMD EPYC 7763 node.

M.5 Schema Guards and Open-Ended Routing

ARKA uses deterministic schema guards only when the prompt exposes a known task schema. These guards choose from route families implied by the prompt format and do not inspect answer labels, model predictions, or benchmark scoring outputs. MemoryAgentBench prompts are schema-covered, so the guard acts as a reliability layer for known benchmark formats. In contrast, LoCoMo and MemBench contain natural conversational-memory questions. Across the evaluated LoCoMo and MemBench rows, schema guards do not fire, and the final top-level route is selected by the learned CPU router.

The intended deployment interpretation is therefore mixed rather than purely hard-coded or purely learned. Known prompt schemas can use deterministic guardrails to avoid invalid route heads. Open-ended memory queries use the learned MLP route. This mirrors production agents, where some workflows have known tool or task schemas while many user queries arrive as natural language.

M.6 Minimal Reproduction Commands

After extracting the anonymous supplement, enter `Supplementary_Material/arka`. The commands below use paths relative to that directory. The CPU-only smoke tests do not require benchmark datasets or backbone weights:

```
cd Supplementary_Material/arka
PYTHONPATH=code python examples/router_smoke.py
PYTHONPATH=code python examples/cpu_memory_demo.py
```

The first command loads the released first-level router and checks that a natural-language query is mapped to a valid ARKA route. The second command builds a small CPU-side memory view and assembles a working context without calling a backbone LLM.

The first-level router can be retrained from the released clean data with:

```
PYTHONPATH=code python code/train_dehm_litelearn_router.py \
  --data_path \
  data/routertrain_clean_r4_no_validation/litelearn_router_train.jsonl \
  --output_root runs/router_retrain \
  --epochs 80 --batch_size 64 \
  --lr 2e-3 --weight_decay 1e-4 --hash_bins 4096 \
  --token_embed_dim 64 --hidden_dim 128 \
  --balanced_loss --selection_metric macro_f1
```

The delegated memory-policy router can be rebuilt and retrained with:

```
PYTHONPATH=code python code/build_clean_memory_policy_dataset.py \
  --output_root data/clean_memory_policy_rebuilt
PYTHONPATH=code python code/train_dehm_policy_router.py \
  --data_path \
  data/clean_memory_policy_rebuilt/policy_router_train.jsonl \
  --output_root runs/memory_policy_retrain \
  --epochs 40 --batch_size 256
```

Full LLM evaluation additionally requires the benchmark datasets and backbone weights. The runner entry points consume local dataset paths and model paths so that benchmark data can remain outside the released artifact.

N Additional Results

Table 8 gives the transfer and deployment rows referenced in Section 5. The same CPU router checkpoint is used across Llama-3.1-8B precision settings and Qwen3-8B. W3 and W4 denote three- and four-bit weight-only precision. Both W4 rows use GPTQ (Frantar et al., 2023). Llama-3.1-8B is quantized by the evaluation runner. Qwen3-8B loads a prequantized checkpoint whose calibration metadata are unavailable in the retained records. The supplementary code's standalone Hugging Face W4 option uses NF4 and does not reproduce these GPTQ rows.

Table 8: Backbone and precision transfer for ARKA. The CPU router is reused without reader-specific retraining. LoCoMo uses the runtime proxy metric in this table, not the main-text QA aggregate. The mean averages different benchmark metrics and is descriptive only.

Backbone / precision	MAB	LongBench-5	LoCoMo proxy	MemBench	Mean
Llama-3.1-8B W3	48.6	53.6	36.8	56.2	48.8
Llama-3.1-8B W4	49.2	55.7	37.1	60.5	50.6
Llama-3.1-8B BF16	50.6	53.7	39.1	62.5	51.5
Qwen3-8B BF16	**53.9**	**56.8**	**38.9**	**67.5**	**54.3**
Qwen3-8B W4	53.3	56.0	34.9	65.7	52.5

Table 9 lists the numeric points used in Figure 3. The scaled-history construction prepends turns from other MemBench dialogues in the same subgroup before the target dialogue, so the answer remains unchanged while unrelated multi-turn history grows.

Finally, schema overrides do not drive the natural conversational-memory rows: on LoCoMo (1,986 queries) and MemBench (1,100 queries), overrides fire on 0.00% of examples, and the final top-level route is always selected by the learned CPU router. MAB prompts are schema-covered, so deterministic schema guards apply there and agree with the raw MLP route on 73.95% of evaluated queries.

Table 9: Length-scale stress-test data for 50 MemBench participation examples. Backbone input is the answer-time input length. Pre-query KV is the live backbone cache before the final question is asked.

Scale	Method	Accuracy	Backbone input	Pre-query KV	History stream	Query time	Total session
1×	Full-context	32.0	2502	2405	2.964 s	0.125 s	3.089 s
2×	Full-context	42.0	4838	4740	6.097 s	0.124 s	6.221 s
5×	Full-context	32.0	11930	11832	17.257 s	0.166 s	17.423 s
10×	Full-context	30.0	23751	23653	44.249 s	0.230 s	44.478 s
50×	Full-context	–	–	–	–	–	OOM
100×	Full-context	–	–	–	–	–	OOM
1×	ARKA	38.0	477	47	0.039 s	0.123 s	0.170 s
2×	ARKA	48.0	477	47	0.039 s	0.115 s	0.168 s
5×	ARKA	42.0	477	47	0.037 s	0.112 s	0.167 s
10×	ARKA	44.0	479	47	0.040 s	0.112 s	0.181 s
50×	ARKA	38.0	479	47	0.039 s	0.118 s	0.322 s
100×	ARKA	36.0	480	47	0.039 s	0.120 s	0.498 s