Q-MEM: QUERY-INFORMED ONLINE MEMORY FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Compact recurrent memory provides LLM agents with a bounded-state mechanism for maintaining information across long-running interactions. However, as interaction histories grow, new evidence can render existing memory associations stale, creating a fundamental tension between revising outdated information and preserving useful memory. To address these challenges, we introduce Q-mem, a query-informed online memory module for frozen decoder language models. Q-mem decouples what to correct from where to update. Specifically, a gated component of the memory query orthogonal to the incoming key refines the selection of old content to revise, while the key remains the sole update address for both correction and insertion. Independent correction and write gates, together with bounded retention, separately control revision, insertion, and persistence. A request-local temporary state further allows the full user query to condition memory access without committing the request to persistent history. Across three backbones, Q-mem achieves the strongest aggregate performance among the compared methods. These results show that query-informed online revision can improve bounded recurrent memory without updating the backbone.

1 INTRODUCTION

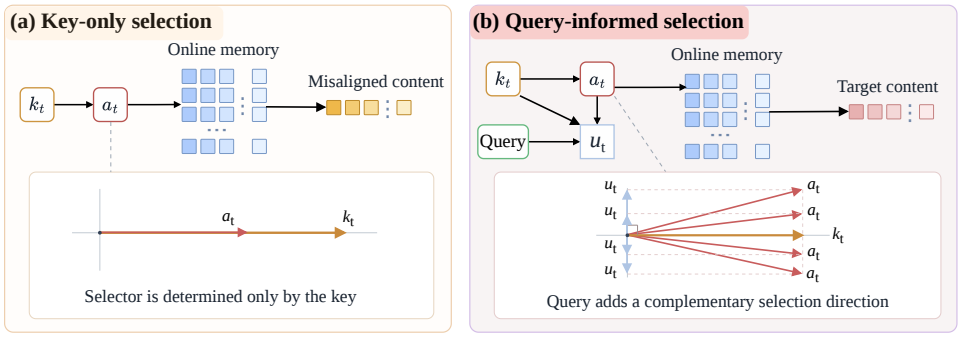

Figure 1: Motivation for query-informed selective revision. (a) Key-only correction selects old content from the incoming key, which can revise the wrong association when the query points elsewhere. (b) Q-mem adds the component of the current query orthogonal to the key to select answer-relevant old content, while retaining the same key as the update address. Thus, Q-mem changes what is corrected without changing where the state is written.

Recent advances in large language models (LLMs) have enabled interactive agents that assist users across repeated conversations and extended tasks (Packer et al., 2023; Park et al., 2023). These applications require agents not only to respond to the current request, but also to retain and reuse information from earlier interactions. As interaction histories grow beyond the active context, maintaining access to useful historical information becomes essential. External memory systems address this need by storing explicit records and retrieving relevant information when needed (Lewis et al., 2020; Zhong et al., 2024). Recurrent latent memory offers a complementary approach: compressing history into a state that directly influences subsequent model computation (Bulatov

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.

et al., 2022; Munkhdalai et al., 2024). In particular, a fixed-size associative state allows an agent to carry information across interactions without expanding its memory storage, making it an attractive approach for long-running applications.

However, maintaining a compact memory requires more than accumulating information. New evidence can make earlier associations outdated, while overwriting the shared state can disrupt information that remains useful. Delta-style associative memories mitigate interference through corrective writing: they retrieve the response associated with an incoming key, subtract a gated version of that response, and write a new value at the same key (Schlag et al., 2021; Yang et al., 2024). In this rule, the key determines both which old content is selected for correction and where the update acts. By contrast, the memory query reads the state for the current computation and is learned indirectly through final-answer supervision, suggesting that it could also inform the selection of content for correction. This raises a natural question: **Can the memory query help determine what content to correct, while the write key continues to determine where the update acts?**

In this paper, we introduce Q-mem, a compact recurrent memory module for a frozen full-attention decoder. Q-mem uses the memory query to refine the old content selected for correction, while retaining the incoming key as the address for both correction and writing. Specifically, the query contributes through its gated component orthogonal to the key, providing a complementary selection direction rather than a second update address. Independent gates control old-content selection and new-value insertion, while a bounded retention operator regulates persistence of the previous state. Furthermore, a request-local recurrence allows the complete user question to condition a temporary copy of memory. Answer generation reads this copy with further writes disabled, and the copy is discarded after the response, leaving persistent history unchanged. The memory module is trained end to end through final-answer supervision, without updating the backbone parameters.

To evaluate Q-mem, we use LoCoMo (Maharana et al., 2024), LongMemEval-S (Wu et al., 2024), and controlled context recovery on HotpotQA and LoCoMo (Yang et al., 2018; Maharana et al., 2024). Experiments cover Qwen3-4B-Instruct, Qwen3-8B, and Llama-3.2-3B-Instruct (Yang et al., 2025; Meta AI, 2024). Across these settings, Q-mem achieves the strongest aggregate LoCoMo performance under the reported F1 and B1 metrics. Structural, context-recovery, and initialization ablations examine the sources and robustness of these gains. Our contributions are the following:

- We formulate query-informed selective revision for compact online agent memory, allowing the memory query to refine which old content participates in correction while preserving the incoming key as the write address.
- We implement this transition as a lightweight adapter for a frozen full-attention decoder, with bounded retention, independently controlled correction and writing, and an explicitly separated request-local temporary state.
- We evaluate the resulting memory across multiple backbones and complementary long-term memory protocols, and use structural, context-recovery, and initialization analyses to identify which transition components account for the observed gains.

2 Related Work

Memory for LLM agents. Memory for LLM agents spans retrieval interfaces, external memory management, and model-side memory modules. Retrieval-augmented generation and lexical retrieval such as BM25 select historical or external text for the current prompt (Lewis et al., 2020; Robertson & Zaragoza, 2009). Agent systems such as MemGPT, Generative Agents, MemoryBank, and Mem0 organize, consolidate, and retrieve persistent records across interactions (Packer et al., 2023; Park et al., 2023; Zhong et al., 2024; Chhikara et al., 2025). Memory-R1 instead learns explicit memory operations over an external store (Yan et al., 2026). Other memory modules encode reusable information in structured or parametric forms: PlugMem provides a task-agnostic plugin memory for agent interactions (Yang et al., 2026), MemoryLLM treats feed-forward parameters as an interpretable token-wise memory (Jaiswal et al., 2026), ParamMem stores reflective patterns in model parameters (Yao et al., 2026), and ReasoningBank records reusable reasoning strategies from prior tasks (Ouyang et al., 2026). These systems keep memory outside the bounded associative state studied here; our transition directly changes the representation used by later computation.

Model-side recurrent memory. Recurrent Memory Transformer propagates learned memory tokens across segments (Bulatov et al., 2022); Infini-attention combines local attention with a compressive recurrent store (Munkhdalai et al., 2024); and Titans learns a neural long-term memory updated at test time (Behrouz et al., 2025). Hebbian and associative agent-memory systems such as HeLa-Mem also use online updates to maintain long-term interaction information (Zhu et al., 2026). Recent agent-oriented approaches further explore compact representations for consolidation and long-horizon reasoning (Zhou et al., 2026; Zhang et al., 2026; Liu et al., 2026). These works establish latent state as a viable carrier of history. Our focus is the transition itself: how an incoming event should revise only the part of a compact associative state implicated by the current request.

Associative online updates. Fast-weight memories interpret a recurrent matrix as an online key–value store (Schlag et al., 2021). Related long-term memory systems either cache past key–value representations for retrieval or learn multi-granularity association and selection over conversational records (Wu et al., 2022; Wang et al., 2023; Xu et al., 2026). Delta-style updates reduce interference by correcting the value retrieved at an incoming key before writing the new target (Yang et al., 2024). Our transition preserves this associative interface but lets the current query alter the corrective content selector, keeps the update support at the incoming key, and learns independent controls for correction and writing. This distinction is central to selective revision: the query changes which old content participates in replacement rather than creating an unconstrained second write address.

3 Q-mem

Q-mem augments a frozen decoder with a bounded associative state. Each token first reads the previous state and injects the resulting memory response into the attention query and output paths. When writes are enabled, the state is updated only after this read. The transition retains part of the old state and selects correction content by augmenting key-based selection with a gated query component orthogonal to the incoming key. Correction and insertion share the same key as their update address but use independent gates. For each request, the persistent history state is copied into a temporary state so that question tokens can condition memory access without becoming persistent history. Answer tokens read this state but do not write to it. Sections 3.1–3.3 define the memory interface, selective revision transition, and request-local recurrence.

3.1 Memory interface

Q-mem augments each adapted layer ℓ of a frozen full-attention decoder with a recurrent state for every memory head h. A memory head is one independent recurrent address–value matrix attached to an adapted layer; it is distinct from a backbone self-attention head. In general, $\mathbf{M}_{\ell,h,t} \in \mathbb{R}^{r_v \times r_k}$ is a linear map from an address space $\mathcal{A} \simeq \mathbb{R}^{r_k}$ to a value space $\mathcal{V} \simeq \mathbb{R}^{r_v}$. We use $r_k = r_v = r$, with r smaller than the hidden width d, and omit ℓ and h except where their distinction matters. Thus, the one-head state below is $\mathbf{M}_t \in \mathbb{R}^{r \times r}$. Only memory parameters ϕ are trained; backbone parameters θ remain fixed. Given $\mathbf{h}_t \in \mathbb{R}^d$, the memory query, update key, and candidate value are

$$\mathbf{q}_t = \mathrm{norm}(\tanh(W_q \mathbf{h}_t)), \qquad \mathbf{k}_t = \mathrm{norm}(\tanh(W_k \mathbf{h}_t)), \qquad \mathbf{v}_t = W_v \mathbf{h}_t, \tag{1}$$

where norm denotes ℓ_2 normalization. $\mathbf{q}_t$ and $\mathbf{k}_t$ lie in the address space, whereas $\mathbf{v}_t$ lies in the value space; all have dimension r in our implementation. Reading the previous state maps the address query to the value response $\mathbf{m}_t \in \mathcal{V}$ and produces two memory-dependent attention residuals:

$$\mathbf{m}_t = \mathbf{M}_{t-1}\mathbf{q}_t, \qquad \Delta\mathbf{Q}_t = \frac{\alpha}{r} W_{\Delta q}\mathbf{m}_t, \qquad \Delta\mathbf{O}_t = \frac{\alpha}{r} W_{\Delta o}\mathbf{m}_t, \tag{2}$$

where α sets the readout scale. The query residual is added before query normalization and rotary position encoding; the output residual is added after the attention output projection. No residual is injected directly into the backbone key or value paths. When writes are enabled, the state is updated only after this read, so a token cannot read its own write; Figure 2 summarizes the architecture.

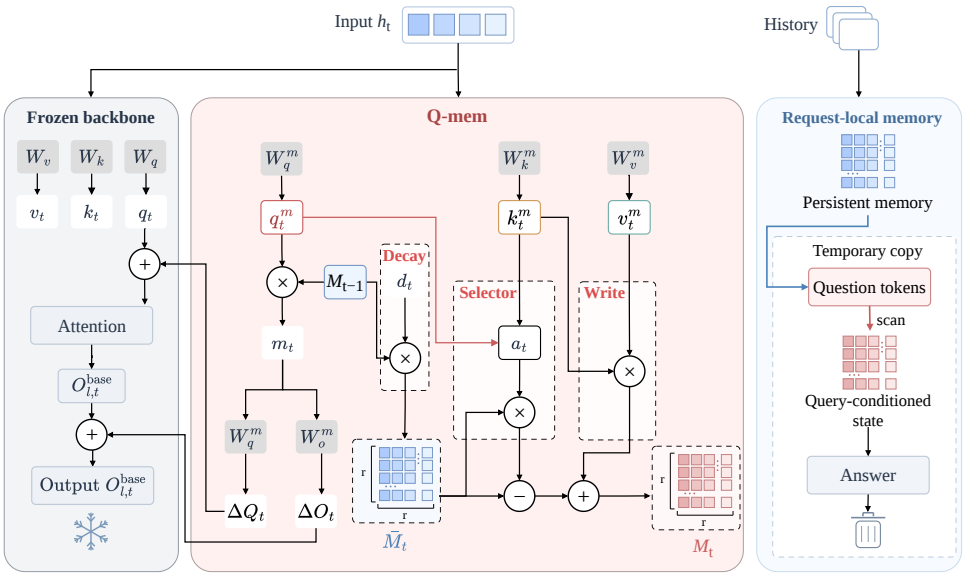

Figure 2: Overview of the Q-mem architecture. Q-mem reads the recurrent state to steer the frozen attention computation and then updates the state through a query-informed, gated read-before-write transition. Persistent history is copied to a request-local temporary state for question-conditioned memory access and discarded after answering.

3.2 Query-informed selective revision

Bounded retention and independent gates. The three token-dependent controllers are computed from the current hidden state, while the retention scale $\mathbf{s}_D$ is a learned coordinate-wise parameter:

$$\begin{aligned} \mathbf{z}_t &= W_D \mathbf{h}_t + \mathbf{b}_D, \qquad \mathbf{d}_t = \exp[-\exp(\mathbf{s}_D) \odot \mathrm{softplus}(\mathbf{z}_t)]\,, \\ \boldsymbol{\gamma}_t &= \sigma(W_\gamma \mathbf{h}_t + \mathbf{b}_\gamma), \quad \boldsymbol{\omega}_t = \sigma(W_\omega \mathbf{h}_t + \mathbf{b}_\omega). \end{aligned} \tag{3}$$

For each adapted layer and memory head, $W_D, W_\gamma, W_\omega \in \mathbb{R}^{r\times d}$, $\mathbf{b}_D, \mathbf{b}_\gamma, \mathbf{b}_\omega \in \mathbb{R}^r$, and $\mathbf{s}_D \in \mathbb{R}^r$ is shared across tokens but not across layers or memory heads. The controller outputs are therefore rank-wise vectors, not one scalar shared by the whole state. Here $\mathbf{d}_t \in (0,1]^r$ controls retention along address coordinates, $\boldsymbol{\gamma}_t \in (0,1)^r$ controls old-content selection, and $\boldsymbol{\omega}_t \in (0,1)^r$ controls insertion in the value space. We form $D_t = \mathrm{Diag}(\mathbf{d}_t)$, $\bar{\mathbf{M}}_t = \mathbf{M}_{t-1} D_t$, and $\tilde{\mathbf{v}}_t = \boldsymbol{\omega}_t \odot \mathbf{v}_t$. Because correction and insertion use different predictors, the model can revise an existing association without forcing the same amount of new content to be written. With corrective selector $\mathbf{a}_t$, the update is

$$\mathbf{M}_t = \bar{\mathbf{M}}_t - (\bar{\mathbf{M}}_t \mathbf{a}_t)\mathbf{k}_t^\top + \tilde{\mathbf{v}}_t \mathbf{k}_t^\top. \tag{4}$$

Here $\mathbf{a}_t \in \mathcal{A}$ selects the old value-space response used for correction, while $\mathbf{k}_t \in \mathcal{A}$ is the sole right-side address for both correction and insertion. The correction uses the retained pre-write state.

Query-informed selector. We augment key-based selection with the component of the gated memory query orthogonal to the normalized key. Since $\mathbf{k}_t$ is ℓ_2-normalized in Equation 1, $I - \mathbf{k}_t\mathbf{k}_t^\top$ is the orthogonal projector onto the complement of the key span:

$$\begin{aligned} \mathbf{u}_t &= (I - \mathbf{k}_t\mathbf{k}_t^\top)(\boldsymbol{\gamma}_t \odot \mathbf{q}_t), \\ \mathbf{a}_t &= \boldsymbol{\gamma}_t \odot \mathbf{k}_t + \eta_{\ell,h}\mathbf{u}_t. \end{aligned} \tag{5}$$

The projection makes $\mathbf{u}_t$ orthogonal to $\mathbf{k}_t$. Its scaled contribution changes the old response $\bar{\mathbf{M}}_t\mathbf{a}_t$ selected for correction. Each layer and memory head has a learned scalar strength:

$$\eta_{\ell,h} = \eta_{\max}\sigma(\vartheta_{\ell,h}). \tag{6}$$

The logit $\vartheta_{\ell,h}$ is trained with the other memory parameters by answer supervision. Substituting Equation 4 gives, for any probe $\mathbf{x}$,

$$(\mathbf{M}_t - \bar{\mathbf{M}}_t)\mathbf{x} = (\widetilde{\mathbf{v}}_t - \bar{\mathbf{M}}_t\mathbf{a}_t)(\mathbf{k}_t^\top \mathbf{x}). \tag{7}$$

Thus, the correction and insertion leave probes orthogonal to $\mathbf{k}_t$ unchanged relative to the retained state: the query changes the selected content, while $\mathbf{k}_t$ remains the sole update address. Disabling retention and the query term and setting $\boldsymbol{\gamma}_t = \boldsymbol{\omega}_t = \beta_t \mathbf{1}$ recovers the standard delta update (Schlag et al., 2021; Yang et al., 2024).

3.3 Request-local recurrence and training

State lifecycle. At the beginning of a conversation, every layer–head state is initialized to zero. Let $C = (c_1, \ldots, c_T)$ denote the sequence of committed history tokens and let Scan_ϕ apply the read in Equation 2 followed by the update in Equation 4 causally. The persistent state is

$$\mathbf{M}^{\mathrm{mem}} = \mathrm{Scan}_\phi(\mathbf{0}, C). \tag{8}$$

At each token, the layer first reads $\mathbf{M}_{t-1}$ and only then, when writing is enabled, produces $\mathbf{M}_t$. Therefore, a token cannot consume its own write. Only committed history changes $\mathbf{M}^{\mathrm{mem}}$. During training, the persistent write mask is one only for the committed history prefix C produced by the episode builder. The explicit recent context and the question turn are not committed by this scan.

Request-local memory. For a user question $Q = (x_1, \ldots, x_s)$, we make a request-local copy of the persistent state and scan only the question tokens:

$$\begin{aligned} \widetilde{\mathbf{M}}^{(0)} &= \mathrm{copy}(\mathbf{M}^{\mathrm{mem}}), \\ \widetilde{\mathbf{M}}^{\mathrm{query}} &= U_\phi^{(s)}(\widetilde{\mathbf{M}}^{(0)}, Q), \end{aligned} \tag{9}$$

where $U_\phi^{(s)}$ applies Equation 4 as the decoder processes the s question tokens. The default request-local write mask selects question tokens only: prompt-visible recent context is read by the frozen decoder but does not update this temporary state, and answer or teacher-forced response tokens are read-only. Answer generation reads this temporary state with further writes disabled. The copy is discarded afterward and $\mathbf{M}^{\mathrm{mem}}$ is restored unchanged. Hence each request has three explicit phases: committed-history writing, temporary query adaptation, and read-only answer generation. The full question can condition memory access without committing either question or answer tokens to persistent history. During training, the copy operation is logical rather than a gradient stop, so final-answer gradients still propagate through the temporary and historical scans.

Answer-supervised training. Each episode contains a historical prefix C, an explicit context X containing the user question, and a target response $Y = (y_1, \ldots, y_n)$. We mix a recent-context-plus-state view, which exposes a bounded recent suffix alongside memory, with a state-reliant view, which moves more of the older evidence into the history prefix and leaves only the immediate query turn explicit. Let z denote a draw from the configured view distribution and let $(C^{(z)}, X^{(z)})$ denote the resulting partition. In the two-view abstraction, the probability of the state-reliant view is denoted by p_{sr}. Scanning $C^{(z)}$ from a zero state and applying the request-local recurrence yields $\widetilde{\mathbf{M}}^{(z)}$. With answer-time writes disabled, we minimize

$$\mathcal{L}(\phi) = -\mathbb{E}_z \left[\frac{1}{n} \sum_{j=1}^{n} \log p_{\theta,\phi}\big(y_j \mid X^{(z)}, y_{<j}, \widetilde{\mathbf{M}}^{(z)}\big) \right]. \tag{10}$$

Both views supervise the same response. Gradients through the history scan train the memory transitions without memory-operation labels. State-reliant examples encourage dependence on information carried by memory. The token-level write masks are part of the training definition. Their episode construction and sampling probabilities are specified in Appendix A.6.

With H heads, persistent state and recurrent-state operations require $O(Hr^2)$ storage per adapted layer and $O(Hr^2)$ work per token, respectively. The projected query adds $O(Hr)$ work. These bounds exclude backbone computation and dense projection/readout costs.

Table 1: Experimental results on the LoCoMo dataset of QA tasks across four categories (Multi Hop, Temporal, Open Domain, and Single Hop) using different methods and three backbones. Results are reported as percentages using F1 and BLEU-1, where higher is better. The "+" prefix denotes a memory method attached to the corresponding backbone. Best performance within each backbone block and metric column is marked in bold, and second-best performance is underlined.

Backbone	Method	Multi-hop		Temporal		Open-domain		Single-hop		Overall	
		F1	B1	F1	B1	F1	B1	F1	B1	F1	B1
Qwen3-4B-Instruct	+BM25 RAG	13.37	8.94	27.16	23.68	8.30	8.03	37.23	30.46	28.96	23.71
	+LLMLingua-2	24.71	19.69	28.41	23.34	8.08	7.44	32.83	26.09	28.88	23.18
	+Context2LoRA	35.18	26.76	35.62	32.21	**17.01**	**15.61**	55.08	48.65	45.01	39.16
	+MLP Memory	30.96	24.40	30.71	26.42	10.62	9.99	47.54	36.86	38.69	30.73
	+δ-mem	33.44	26.12	35.64	30.78	12.55	11.24	52.29	43.17	42.89	35.47
	+Q-mem	**38.54**	**29.55**	**38.35**	**34.81**	13.80	12.53	**56.40**	**49.23**	**46.71**	**40.33**
Qwen3-8B	+BM25 RAG	14.87	9.96	30.03	25.62	8.81	7.74	41.84	34.56	32.38	26.52
	+LLMLingua-2	33.12	25.33	29.57	23.56	11.78	10.61	42.84	36.31	36.36	30.04
	+Context2LoRA	37.12	26.40	32.03	27.16	**20.81**	**15.95**	56.45	49.78	45.60	38.67
	+MLP Memory	39.13	**30.98**	31.03	24.50	14.03	12.34	57.41	47.18	45.86	37.31
	+δ-mem	37.34	26.90	35.49	29.57	13.69	11.12	55.54	49.04	45.42	38.56
	+Q-mem	**39.45**	29.96	**35.80**	**30.79**	17.69	14.82	**59.53**	**53.08**	**48.30**	**41.82**
Llama-3.2-3B-Instruct	+BM25 RAG	16.56	11.71	24.05	17.72	12.92	11.13	41.83	32.42	31.69	24.23
	+LLMLingua-2	27.44	22.96	19.21	15.46	11.44	10.33	35.33	29.10	29.04	23.96
	+Context2LoRA	30.53	14.86	26.35	18.41	**18.18**	**15.66**	47.41	38.88	38.11	28.77
	+MLP Memory	31.38	**25.18**	23.83	18.89	11.38	10.90	49.30	38.08	38.35	30.02
	+δ-mem	30.36	15.07	22.16	15.02	17.31	14.44	45.31	36.07	36.00	26.49
	+Q-mem	**32.10**	17.38	**27.09**	**20.49**	14.36	13.02	**50.58**	**41.20**	**40.04**	**30.77**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Dataset and evaluation metrics. We evaluate long-term conversational memory on LoCoMo (Maharana et al., 2024), using 1,540 questions from categories 1–4. These questions test the retrieval of facts from a single session (single-hop), the integration of evidence across sessions (multi-hop), reasoning over time-related cues (temporal), and the combination of conversational evidence with general knowledge (open-domain). We report Mem0-style unique-token F1 and smoothed sentence BLEU-1 (B1) (Chhikara et al., 2025; Papineni et al., 2002) for each question category and overall. Answers are limited to 50 new tokens, and all scores are percentages.

Compared methods. We compare Q-mem with representative memory mechanisms. For textual memory, BM25 RAG (Lewis et al., 2020; Robertson & Zaragoza, 2009) retrieves relevant past turns and prepends them to the answer prompt, while LLMLingua-2 (Pan et al., 2024) compresses conversation history into a shorter textual context. For parametric memory, Context2LoRA (Hu et al., 2022; Back et al., 2026) uses trained low-rank adapters to encode context-dependent information. For outside-channel memory, MLP Memory (Wei et al., 2026) uses an auxiliary memory module separate from the backbone. We also include δ-mem (Lei et al., 2026), which writes history into a compact online associative state.

Implementation details. We evaluate Q-mem across three backbone LLMs: Qwen3-4B-Instruct, Qwen3-8B (Yang et al., 2025), and Llama-3.2-3B-Instruct (Meta AI, 2024). Q-mem keeps the backbone parameters frozen. It uses one rank-8 state head per adapted attention layer and modifies the query and output paths. We train Q-mem for one epoch with seed 42 and four recent messages in the explicit episode context. Q-mem uses an FP32 recurrent state and request-local query recurrence. Its write-sparsity penalty is zero, and the initial bounded query strength is $\eta_0 = 0.067$. We evaluate

Q-mem on LoCoMo with seed 42 and stochastic decoding at temperature $T = 0.4$, top-$p = 0.9$, and top-$k = 10$. More training setup and evaluation configurations are listed in Appendix A.6.

4.2 Main Results

Table 1 compares six memory methods on LoCoMo across three frozen backbones. Paired scores below are reported as F1/B1. Q-mem obtains the highest overall F1 and B1 within every backbone block, with gains over the strongest baseline ranging from 1.69 to 2.44 F1 points and from 0.75 to 3.15 B1 points. These comparisons indicate that the overall gains are retained across the two Qwen3 scales and the Llama model family. Because every block uses the same six memory interfaces, category definitions, and evaluation metrics, this cross-backbone consistency is not explained by a change in task composition. Instead, query-informed revision transfers across scales and model families, with some category-specific backbone variation.

Against the online associative baseline δ-mem, Q-mem improves overall performance by 2.88–4.04 F1 points and 3.26–4.86 B1 points across the three backbones. Both textual-memory baselines have lower overall scores than the other compared methods on each backbone, while Context2LoRA and MLP Memory provide the strongest competing overall results. The gains are not uniform across question types. Q-mem leads temporal and single-hop questions under both metrics, and multi-hop questions under F1, on all three backbones. Context2LoRA instead leads open-domain questions under both metrics throughout. This category-specific gap suggests that open-domain questions benefit more from broad integration of conversational evidence with pretrained world knowledge than from compact associative revision alone, whereas Q-mem is especially well matched to updating and preserving conversational associations. This pattern suggests complementary strengths: query-conditioned revision favors updates, whereas open-domain questions benefit from evidence aggregation.

5 Analysis

We analyze Q-mem from four perspectives: knowledge-update performance across memory interfaces, state utility when explicit history is unavailable, transition-component effects, and sensitivity to query-strength initialization. All analyses use Qwen3-4B-Instruct and the shared configurations in Section 4.1; only context recovery uses a modified input protocol.

5.1 Can Q-mem Improve Knowledge Updating by Selecting What to Revise?

Takeaway. Q-mem demonstrates strong knowledge-updating capability by adapting its answers when conversational information changes.

Figure 3 compares GPT-4o judge accuracy on the LongMemEval-S knowledge-update questions across the six methods. Q-mem reaches 67.95%, ahead of δ-mem (65.38%) and MLP Memory (64.10%). Context2LoRA, BM25 RAG, and LLMLingua-2 obtain 55.13%, 50.00%, and 33.33%, respectively.

Q-mem has a substantial advantage on knowledge-update questions over the other compared methods. The result indicates that it can distinguish an outdated association from newly supplied information that conflicts with it, then revise the stale content while retaining compatible facts. This behavior is consistent with query-informed selection: updates are directed toward the old response that matters for the current question rather than applied uniformly to the entire memory state.

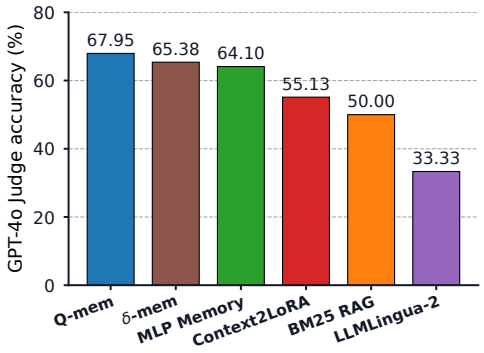

Figure 3: Knowledge-update accuracy on LongMemEval-S, judged by GPT-4o.

5.2 Can Q-mem Recover Answer-Relevant Information without Explicit History?

Takeaway. Q-mem's recurrent state preserves answer-relevant information that remains useful without explicit history, even when request-local query recurrence is disabled.

We test state utility after removing explicit history. The no-context backbone and Q-mem receive the same instruction and question; Q-mem's state is built from history only, with request-local recurrence and answer-time writes disabled. Figure 4 shows recovery on both benchmarks: overall EM/F1 rises from 5.91/6.75 to 13.40/18.31 on HotpotQA, and F1/BLEU-1 from 7.00/5.67 to 10.08/8.35 on LoCoMo. The strongest category-level gains occur on comparison and single-hop questions, respectively. Because the answer-time inputs are otherwise matched, this comparison isolates information carried by the persistent state rather than the benefit of exposing more text to the decoder. Together, these gains show that Q-mem's recurrent state can carry answer-relevant historical information into the answer computation even when the original sessions are not visible.

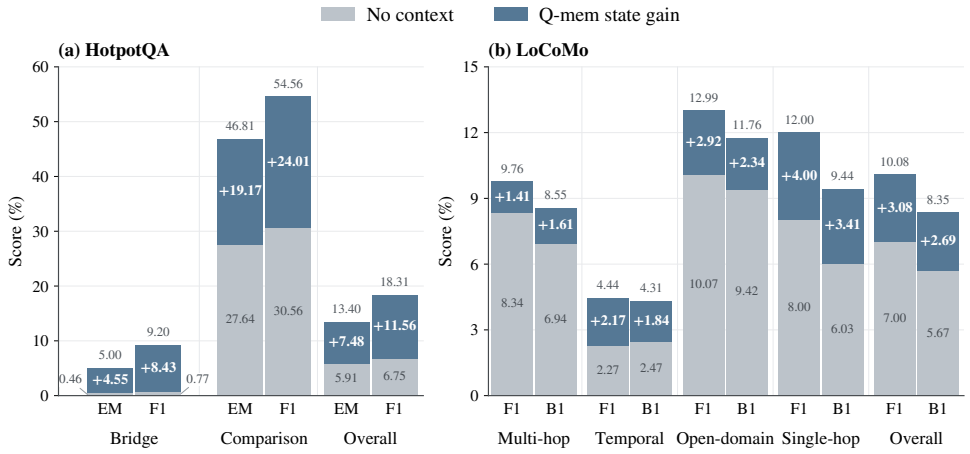

Figure 4: Context recovery on HotpotQA and LoCoMo with Qwen3-4B-Instruct. Gray segments show the no-context backbone scores, while blue segments show the absolute gains from Q-mem state conditioning. The total bar height gives the state-conditioned score. HotpotQA reports EM and F1; LoCoMo reports F1 and BLEU-1 (B1). All scores are percentages, and the two panels use different vertical scales.

5.3 Which Transition Components Account for Q-mem's Gains?

Takeaway. An additional correction direction helps even when it is random, while query dependence yields a further gain. Orthogonal projection, independent correction and write gates, and request-local query recurrence further improve aggregate performance.

Table 2 isolates the transition components under the same training and history-replay protocol. Removing query-informed selection or request-local recurrence lowers aggregate scores; a fixed random correction direction recovers part of the loss, while making it query-dependent adds 0.45 F1 and 0.37 B1 points. Removing the orthogonal projection trades a small aggregate drop for higher open-domain scores, and tying the correction/write gates is the weakest variant. The full transition gives the best overall balance. The random-direction control shows that an additional correction subspace is useful even before its direction is conditioned on the query. The projection and gate results further indicate that selecting a revision target and controlling correction and insertion separately are complementary rather than interchangeable operations.

Table 2: Structural ablations on Qwen3-4B-Instruct evaluated on LoCoMo. All scores are percentages. Bold and underline indicate the best and second-best score in each metric column, respectively.

Variant	Multi-hop		Temporal		Open-domain		Single-hop		Overall	
	F1	B1	F1	B1	F1	B1	F1	B1	F1	B1
w/o Query-Informed Selection	36.05	27.92	37.48	33.66	12.31	10.02	54.85	47.31	45.13	38.59
w/o Request-Local Query Recurrence	36.52	26.01	36.81	33.32	12.62	10.50	54.23	47.38	44.76	38.24
w/o Query-Dependent Direction	<u>36.58</u>	<u>28.70</u>	37.85	33.98	13.65	12.22	**56.44**	<u>49.19</u>	<u>46.26</u>	<u>39.96</u>
w/o Orthogonal Projection	35.58	27.06	<u>38.15</u>	<u>34.32</u>	**14.37**	**12.93**	55.70	48.39	45.78	39.34
w/o Independent Correction/Write Gates	36.01	27.60	37.67	33.78	13.23	11.73	54.25	46.65	44.90	38.30
Q-mem **(Full)**	**38.54**	**29.55**	**38.35**	**34.81**	<u>13.80</u>	<u>12.53</u>	<u>56.40</u>	**49.23**	**46.71**	**40.33**

5.4 How Sensitive Is Q-mem to Initialization Choices?

Takeaway. Stronger query initialization is not always better: the largest tested query-strength value underperforms the no-query reference. Intermediate query-strength and decay-bias initializations yield the best overall performance in our sweeps.

Figure 5 varies the initial query strength η_0 and decay-bias initialization b_D one at a time under the fixed protocol. The query-strength sweep is non-monotonic: the default $\eta_0 = 0.067$ reaches 46.71/40.33 F1/B1, whereas 0.100 falls to 43.30/36.50, below the zero-strength reference; 0.030 gives 45.99/39.65. The decay-bias sweep is flatter but peaks at $b_D = -7$ (46.71/40.33), compared with 44.50/38.23 at -9 and 46.37/39.93 at -5. Thus, initialization controls the revision–retention balance: moderate query strength supports useful correction, while excessive strength can destabilize the learned transition.

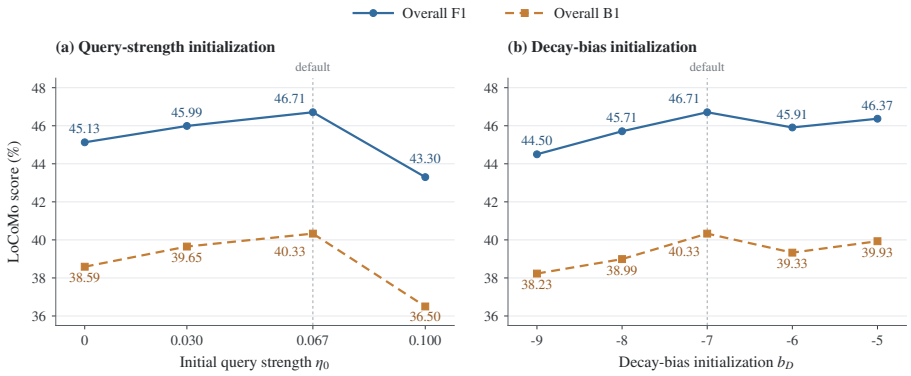

Figure 5: Sensitivity to initialization choices on LoCoMo with Qwen3-4B-Instruct. **Left:** varying the initial query strength η_0. **Right:** varying the decay-bias initialization b_D. Both panels report overall F1 and B1 while varying one initialization and holding the remaining settings fixed. Vertical dashed lines mark the default values used in the main experiments ($\eta_0 = 0.067$ and $b_D = -7$).

6 Conclusion

We presented Q-mem, a query-informed online memory module for frozen decoder language models. Its query-conditioned transition separates stale-content selection from key-based writes, while orthogonal selection, independent gates, bounded retention, and request-local state revise associations without changing backbone parameters. Across LoCoMo and three backbones, Q-mem achieves the strongest aggregate F1 and BLEU-1. Context recovery and structural ablations support the roles of state preservation and selective revision. These findings suggest that combining query-conditioned revision with a bounded recurrent state is a promising direction for online memory.

AI USE STATEMENT

In this work, generative AI tools were used to assist with language editing and improving the readability of the manuscript. All AI-assisted content was reviewed and revised by the authors. The authors take full responsibility for the final content of this work.

ETHICS STATEMENT

Q-mem aims to further advance the field of long-term conversational memory for LLM agents by enabling query-informed online revision in a bounded recurrent state. Our intended use is to advance robust, efficient, long-horizon conversational memory and knowledge updating in open research environments and practical systems. Our experiments use publicly available training corpora and benchmarks. Given this, we foresee no issues regarding fairness, privacy, or security, or any other harmful societal or ethical implications in general.

REPRODUCIBILITY STATEMENT

The method equations, token-level write masks, and controller shapes are given in Section 3 and Appendix A; the complete state lifecycle and transition pseudocode are in Appendix A.5. Appendix A.6 records the training and evaluation protocols, initialization and optimizer treatment, decoding settings, and recurrent state precision. The anonymized implementation is available at `https://anonymous.4open.science/r/Q-mem/` and provides the code and configuration needed to reproduce each reported result.

REFERENCES

Seungju Back, Dongwoo Lee, Naun Kang, Taehee Lee, S. K. Hong, Youngjune Gwon, and Sungjin Ahn. Understanding LoRA as knowledge memory: An empirical analysis. *arXiv preprint arXiv:2603.01097*, 2026. URL https://arxiv.org/abs/2603.01097.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2501.00663.

Aydar Bulatov, Yury Kuratov, and Mikhail S. Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, 2022.

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

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, 2021. doi: 10.18653/v1/2021.naacl-main.365.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Ajay Jaiswal, Lauren Hannah, Han-Byul Kim, Duc Hoang, Arnav Kundu, Mehrdad Farajtabar, and Minsik Cho. MemoryLLM: Plug-n-play interpretable feed-forward memory for transformers. *arXiv preprint arXiv:2602.00398*, 2026. URL https://arxiv.org/abs/2602.00398.

Jingdi Lei, Di Zhang, Junxian Li, Weida Wang, Kaixuan Fan, Xiang Liu, Qihan Liu, Xiaoteng Ma, Baian Chen, and Soujanya Poria. δ-mem: Efficient online memory for large language models. *arXiv preprint arXiv:2605.12357*, 2026. URL https://arxiv.org/abs/2605.12357.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, 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.

Zhichen Liu, Ruihan Sun, Hengjie Yang, Zipeng Wu, Zhaohan Chen, Xiaofan Zhang, and Yang Xu. Livemem: Maintaining memory state continuity in long-running llm inference. *arXiv preprint arXiv:2608.02515*, 2026. URL https://arxiv.org/abs/2608.02515.

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*, pp. 13851–13870, 2024. doi: 10.18653/v1/2024.acl-long.747.

Meta AI. Llama 3.2 model card, 2024. URL https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md.

Tsendsuren Munkhdalai, Manaal Faruqui, and Siddharth Gopal. Leave no context behind: Efficient infinite context transformers with infini-attention. *arXiv preprint arXiv:2404.07143*, 2024.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25140.

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

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*, pp. 963–981. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.57. URL https://aclanthology.org/2024.findings-acl.57/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, 2002. doi: 10.3115/1073083.1073135.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

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.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, 2021.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2306.07174.

Rubin Wei, Jiaqi Cao, Jiarui Wang, Jushi Kai, Qipeng Guo, Bowen Zhou, and Zhouhan Lin. MLP memory: A retriever-pretrained memory for large language models. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d72ae75abaa70a3b19c5d4f436c680d1-Abstract-Conference.html.

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

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2203.08913.

Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Wenlin Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, and Tong Xu. From single to multi-granularity: Toward long-term memory association and selection of conversational agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=i2yIvZARnG.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

An Yang et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

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*, pp. 2369–2380, 2018.

Tianjun Yao, Yongqiang Chen, Yujia Zheng, Pan Li, Zhiqiang Shen, and Kun Zhang. ParamMem: Augmenting language agents with parametric reflective memory. *arXiv preprint arXiv:2602.23320*, 2026. URL https://arxiv.org/abs/2602.23320.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vI56m4Iu4e.

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.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XY8AaxDSLb.

Jinchang Zhu, Jindong Li, Cheng Zhang, Jiahong Liu, and Menglin Yang. HeLa-Mem: Hebbian learning and associative memory for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13757–13769. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.625. URL https://aclanthology.org/2026.acl-long.625/.

A Additional Details of Q-mem

A.1 Controller Properties

Equation (3) gives the complete controller parameterization used in the main method. Since $\mathrm{softplus}(z_t) \geq 0$ and $\exp(s_D) > 0$, every retention coordinate lies in $(0, 1]$. This non-amplification property applies to retention alone, not to the complete state update. The correction gate acts in the address space, whereas the write gate acts in the value space; their independence does not require the resulting correction and insertion vectors to have equal norms.

A.2 Equivalent Transition and Local Query Effect

Writing

$$c_t = \bar{M}_t a_t, \qquad \tilde{v}_t = \omega_t \odot v_t, \tag{11}$$

Equation (4) has the equivalent form

$$M_t = M_{t-1} D_t \left(I - a_t k_t^\top\right) + \tilde{v}_t k_t^\top. \tag{12}$$

The query component can be evaluated without constructing the projection matrix. Let

$$g_t = \gamma_t \odot q_t, \qquad u_t = g_t - k_t(k_t^\top g_t). \tag{13}$$

This requires $O(r)$ work per head.

To isolate the query term locally, hold the previous state and all other controller outputs fixed and compare the transition with strength $\eta_{\ell,h}$ against the same transition with strength zero. With $\chi_t = k_t^\top q_t$, the difference when probing the updated states at q_t is

$$\Delta m_t^{\mathrm{query}} = [M_t(\eta_{\ell,h}) - M_t(0)]\, q_t = -\eta_{\ell,h} \chi_t \bar{M}_t u_t. \tag{14}$$

It vanishes when the query and key are orthogonal or when the gated query is collinear with the key. This is a post-update probe comparison, not the actual read at token t, which uses M_{t-1}.

Disabling retention ($D_t = I$) and the query term ($\eta_{\ell,h} = 0$), and setting $\gamma_t = \omega_t = \beta_t \mathbf{1}$, reduces Equation (4) to

$$M_t = M_{t-1} - \beta_t \left(M_{t-1} k_t - v_t\right) k_t^\top, \tag{15}$$

which is the standard corrective associative update (Schlag et al., 2021; Yang et al., 2024).

A.3 Numerical State

The reported main configuration uses a single memory head with $r = 8$. Recurrent matrices are maintained in FP32 while the frozen backbone operates in bf16; reads are cast back to the backbone dtype for subsequent computation. This changes accumulation precision without changing the transition. Persistent memory size does not include the trainable controller parameters, the backbone key–value cache, or training activations.

A.4 Parameter Shapes and Sharing

For every adapted layer ℓ and memory head h, the projection matrices W_q, W_k, and W_v have shape $r \times d$, where d is the hidden width and r is the recurrent address/value rank. The retention and gate matrices W_D, W_γ, and W_ω also have shape $r \times d$; their biases and the learned retention scale s_D have shape r. The query-strength logit $\vartheta_{\ell,h}$ is one scalar per layer and memory head, whereas γ_t, ω_t, and d_t are rank-wise vectors. No controller or strength logit is shared across adapted layers or memory heads. The reported implementation attaches one recurrent memory head to each adapted attention layer and uses $r = 8$; the backbone's native self-attention heads are not memory heads.

A.5 State-Lifecycle Pseudocode

The following procedure specifies the order of reads, writes, and resets used by both training and evaluation. "Read" denotes Equation (2) and "Write" denotes Equations (3)–(4).

1. At the start of an episode, set every layer–head state to zero.
2. For each token in the committed history prefix C, read the current state and then write the token update. Store the resulting state as M^{mem}.
3. For a request, copy M^{mem} to a temporary state. Process the question tokens causally, reading before each update. Under the main question-only scope, only question-token positions have a temporary write mask.
4. Generate the answer while reading the temporary state with all memory writes disabled. Answer and teacher-forced response tokens are therefore read-only.
5. Discard the temporary state and restore the persistent state before the next request. A question or answer never becomes committed history unless a separate evaluation protocol explicitly treats it as a new conversation turn.

This order enforces causal non-self-reading: a token can affect later tokens, but cannot read the write produced at its own position. The persistent write mask is applied only to committed history; prompt-visible recent context is available to the frozen decoder but is not silently added to the persistent or request-local state under the main scope.

A.6 Training and Evaluation Reproducibility

This section records the experimental protocol used for the reported LoCoMo comparisons, Context Recovery analysis, structural ablations, and initialization sensitivity studies.

A.6.1 Training Configuration

We train the trainable memory methods on the same 2,219 QASPER episodes (Dasigi et al., 2021) and evaluate on the 10 conversations in LoCoMo (Maharana et al., 2024). The reported LoCoMo split contains 1,540 questions from categories 1–4: 282 multi-hop, 321 temporal, 96 open-domain, and 841 single-hop questions. The three frozen decoder backbones are Qwen3-8B and Qwen3-4B-Instruct (Yang et al., 2025), and Llama-3.2-3B-Instruct (Meta AI, 2024). The comparison across all three backbones appears in Table 1.

For each QASPER episode, the answer is the sole supervised target. A historical prefix is processed before the current request, while four recent messages remain in the explicit episode context in the principal training view. A state-reliant view places more preceding evidence in the memory-write prefix and leaves the immediate request explicit. The state-reliant view is sampled with probability 0.2. Both views use the same target answer; no labels specifying when to retain, erase, or write a memory association are provided. The backbone parameters remain frozen and gradients update only the method-specific trainable module. For Q-mem, the temporary request-local recurrence is part of the answer-supervised computational path and is enabled in the main LoCoMo evaluation. Answer tokens are generated with further memory writes disabled.

Unless a method has no training phase, experiments use one QASPER training epoch, seed 42, data seed 42, and four recent messages. The recurrent Q-mem run uses token-level state writing, a rank-8 memory, and one state head per adapted attention layer. Table 3 records the principal optimizer and sequence settings of that run.

Q-mem uses a rank-8 state with one head per adapted layer, query- and output-path residuals, and a readout scale of $\alpha = 16$. The retention-bias and log-scale parameters are initialized to -7 and -1, respectively; the correction- and write-gate biases are each initialized to -1. The query-informed correction strength is $\eta_{\ell,h} = 0.25\,\sigma(\vartheta_{\ell,h})$, with $\vartheta^{(0)}_{\ell,h} = -1$, so its initial value is $\eta_0 = 0.25\sigma(-1) \approx 0.067235$. Its trainable logit has no weight decay and receives a learning-rate multiplier of 10 relative to the other memory parameters. The query-informed correction direction is the gated query after orthogonal projection against the incoming key. The main run uses FP32 recurrent state, request-local query recurrence, and zero write-sparsity penalty. The frozen backbone executes in bf16; the state precision changes recurrent accumulation rather than the mathematical update rule.

Setting	Value
Training set	2,219 QASPER episodes
Training epochs	1
Recent explicit messages	4
Seed / data seed	42 / 42
Maximum read length	512 tokens
Maximum write length	8,192 tokens
Per-device training batch size	1
Distributed processes	4
Gradient accumulation steps	8
Effective global batch size	32
Learning rate	2×10^{-4}
Scheduler / warmup ratio	cosine / 0.10
Weight decay	0
Optimizer	fused AdamW
Attention implementation	FlashAttention-2
Distributed optimizer	DeepSpeed ZeRO-2
Frozen-backbone computation	bf16
Q-mem recurrent state	FP32

Table 3: Training configuration of the principal Q-mem run. The one-epoch, four-recent-message, and seed settings are shared with the matched trainable comparisons; method-specific modules may differ.

A.6.2 LoCoMo metric computation

The reported F1 and B1 are the Mem0-style metrics (Chhikara et al., 2025). For F1, the prediction and reference are lowercased; periods, commas, exclamation marks, and question marks are replaced by spaces; whitespace tokenization is applied; and duplicate tokens are removed by converting each token list to a set. Writing the two resulting sets as P and R, we compute

$$p = \frac{|P \cap R|}{|P|}, \qquad r = \frac{|P \cap R|}{|R|}, \qquad \mathrm{F1} = \frac{2pr}{p+r}, \tag{16}$$

with score zero when either token set is empty or when $p + r = 0$. B1 is sentence-level BLEU-1 (Papineni et al., 2002) on lowercased NLTK word tokens, with weights $(1, 0, 0, 0)$ and smoothing method 1. We first compute F1 and B1 for each question, then average the question scores within each category. Overall is the unweighted mean over the 1,540 individual question scores, not the arithmetic mean of the four category means. All reported values are multiplied by 100 to obtain percentages.

A.6.3 Structural-ablation interventions

Every structural ablation is retrained on the same QASPER episodes with one epoch, four recent messages, and seed 42, then evaluated on the same LoCoMo history-replay protocol as full Q-mem. The controlled interventions are as follows.

A.6.4 Fixed versus learned query strength

Table 5 compares the fixed-strength variant with learned query strength under two learning-rate multipliers. All three runs use the same QASPER training set, one epoch, four recent messages, and the same LoCoMo protocol. The fixed variant keeps the bounded strength at its initial value $\eta_0 = 0.067235$, whereas the learned variants optimize the strength controller with a multiplier of 1 or 10 relative to the base learning rate. Scores are the Mem0-style F1 and BLEU-1 values reported for the four LoCoMo categories and their macro aggregate.

Variant	Change relative to full Q-mem
w/o Query-Informed Selection	Set the query-informed correction strength to zero, removing the additional query-derived correction term.
w/o Request-Local Query Recurrence	Do not scan a temporary copy of the state over the current request.
w/o Query-Dependent Direction	Replace the query-dependent correction direction with a fixed random direction.
w/o Orthogonal Projection	Use the gated query without removing its component parallel to the incoming key.
w/o Independent Correction/Write Gates	Tie the correction and insertion controllers instead of learning independent gates.

Table 4: Controlled changes in the structural ablations. Except for the stated intervention, the training and LoCoMo evaluation setup is held fixed.

Variant	Multi-hop		Temporal		Open-domain		Single-hop		Overall	
	F1	B1	F1	B1	F1	B1	F1	B1	F1	B1
Fixed strength ($\eta_0 = 0.067$)	36.87	27.81	36.61	32.92	12.95	11.32	56.22	49.14	45.89	39.50
Learned strength (LR $\times 1$)	36.67	27.39	36.00	31.96	11.33	10.92	55.39	47.99	45.17	38.57
Learned strength (LR $\times 10$)	38.54	29.55	38.15	34.32	13.23	11.73	56.40	49.23	46.64	40.18

Table 5: Query-strength learning ablation on Qwen3-4B-Instruct evaluated on LoCoMo. The learned-strength learning-rate multiplier is relative to the base optimizer rate; all scores are percentages.