RECALL WHAT IS MISSING: COMPOSING EVIDENCE FROM LONG-TERM AGENT MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Under a limited context budget, a long-term agent must recall not merely memories that are individually relevant, but evidence that is collectively sufficient for the current query. Conventional memory access retrieves, scores, and truncates memories, often spending multiple context slots on redundant facts while leaving complementary evidence behind. We cast the read phase of long-term memory as *query-conditioned evidence composition*: each memory is valued by what it adds beyond the evidence already selected. We instantiate this operation with MEMPURSUIT, a training-free mechanism that uses the residual of a sparse convex query reconstruction as a representation-space proxy for what the current memory set is missing. By reusing the query and candidate embeddings already available from retrieval, MEMPURSUIT progressively composes nested evidence contexts without invoking a language model or cross-encoder to score each query–memory pair. The effectiveness of our method is verified on 3 popular benchmarks in comparison to strong baselines. In fixed-budget LoCoMo experiments, MEMPURSUIT improves end-to-end accuracy by 7.2 percentage points at Top-10 and 5.5 percentage points at Top-20, with consistent gains across multiple memory frameworks, while achieving a 32.9× speedup in median reranking latency over fully batched BGE-v2-m3.

1 INTRODUCTION

Long-lived language-model agents accumulate far more interaction history than can be placed in the prompt of every future query. External memory systems address this mismatch by extracting, organizing, and retrieving persistent records of past interactions (Zhong et al., 2024; Chhikara et al., 2025; Kang et al., 2025; Xu et al., 2025). Yet storage alone does not determine whether an agent remembers successfully. At answer time, a memory system must assemble a limited context from its growing store of records. We refer to this process as memory reading. Hierarchical and graph-based architectures exploit the organization of stored memories to guide retrieval (Kang et al., 2025; Gutiérrez et al., 2024; Rasmussen et al., 2025). A common memory-read policy selects records according to their individual relevance to the query, using dense semantic similarity, sparse lexical matching, or hybrid retrieval (Zhong et al., 2024; Pan et al., 2025; Fang et al., 2026), optionally followed by a relevance reranker (Li et al., 2026).

For the dominant pointwise memory-read, the final context consists of the K highest individual scores, implicitly treating the usefulness of a memory as independent of the other selected memories (Sun et al., 2026). In long histories, independently relevant memories often repeat the same event, preference, or conversational fact (Hu et al., 2026). A multi-hop query can consequently receive several paraphrases of one hop while missing a less similar fact required to complete the evidence chain. Figure 1 shows this pattern: the standard read repeats memories about Deborah's father while omitting her friend Karlie, whereas MEMPURSUIT selects complementary records about her father, mother, and Karlie to complete the answer. Increasing K only partially addresses this failure: it consumes more context, exposes the reader to more distractors, and weakens the purpose of a selective memory layer (Ji et al., 2026). The key decision is not just which memories match the query, but which memory adds what the current context still lacks.

We formulate this read-time decision as *query-conditioned evidence composition*: assembling a compact context by valuing each memory according to what it adds to the evidence already selected. Graph-based systems already support cross-memory retrieval through explicit relational

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.

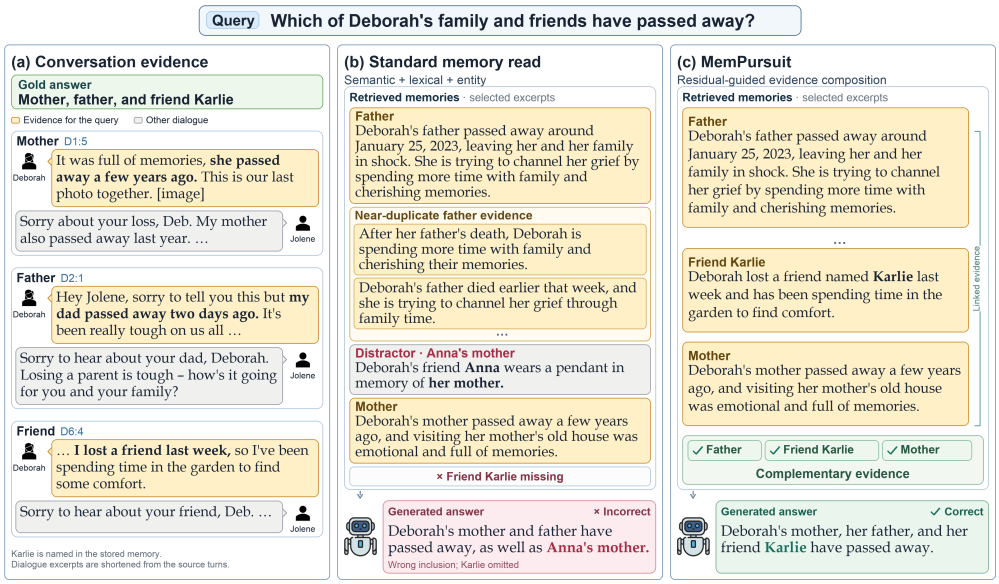


Figure 1: A memory-reading example from LoCoMo. The query asks which of Deborah's family and friends have passed away. The standard read returns repeated records about her father and a distractor about Anna's mother, while omitting her friend Karlie. MEMPURSUIT selects complementary records about her father, mother, and Karlie, enabling the complete answer shown. Dialogue and memory excerpts are shortened for presentation. Additional case studies appear in Appendix C.3.

structures (Gutiérrez et al., 2024; Rasmussen et al., 2025). However, building and maintaining these structures requires additional operations, with graph-wide consolidation identified as a substantial source of overhead as memory grows (Zhou et al., 2026).

We introduce MEMPURSUIT, a training-free reader that implements this principle through sparse convex query reconstruction. Its central idea is to represent the query using the selected memory embeddings and use the remaining residual to guide further recall. Starting with the strongest semantic match, MEMPURSUIT alternates between selecting a memory aligned with the residual and refitting the weights of all selected memories. The residual discounts query directions already represented in the context, directing subsequent selections toward complementary content. A single pursuit path yields nested evidence sets across context budgets. By operating directly on existing query and memory embeddings, MEMPURSUIT makes these conditional choices through matrix and vector operations, without additional language-model or cross-encoder scoring.

In controlled comparisons on LoCoMo, with the memory store, candidate pool, and downstream evaluation fixed, MEMPURSUIT improves Top-10 accuracy by 7.2 percentage points, including a 12.8-point gain on multi-hop questions. It improves gold-session coverage and reaches 85.1% accuracy with 20 memories, outperforming hybrid retrieval with 50. At Top-20, it comes within 0.5 points of BGE-large with 32.7× lower median selection latency. These benefits transfer across four memory frameworks, and its end-to-end pipeline achieves the highest aggregate scores among the evaluated systems on LoCoMo, LongMemEval-S, and BEAM with both answerers. These results establish residual-guided composition as an effective, efficient, and reusable operation for long-term agent memory. Our contributions are threefold:

- **Residual-guided evidence composition.** We introduce MEMPURSUIT, a training-free memory reader that turns query reconstruction into a set-conditioned selection rule. Residual scoring and fully corrective simplex refitting adapt each choice to the evidence already selected.
- **More complete evidence under limited budgets.** Controlled comparisons show higher answer accuracy with fewer memories and improve complete gold-session coverage from 77.1% to 84.5% at Top-10. Reconstruction analysis connects these evidence gains to the geometric selection signal.
- **Reusable and efficient memory reading.** Experiments span three benchmarks and two answerers, with accuracy gains in all eight framework–budget configurations in cross-framework comparisons. Reranker comparisons demonstrate competitive answer quality at a fraction of the selection cost.

2 PRELIMINARIES

2.1 WRITING MEMORIES, READING EVIDENCE

An agent memory system performs two complementary operations. *Writing* constructs and updates a persistent memory store $\mathcal{M} = \{m_i\}_{i=1}^{N}$ from past interactions. *Reading* assembles stored information into the answer model's context for a query q. The writer determines what the agent retains. The reader chooses which memories to use together for the current answer.

For each query, the underlying memory system supplies a candidate pool $\mathcal{C}_P(q) \subseteq \mathcal{M}$ of P memories. A read policy π selects an evidence set $S_K(q) = \pi(q, \mathcal{C}_P(q), K) \subseteq \mathcal{C}_P(q)$ with $|S_K(q)| \leq K$. The selected memory texts are passed to the answer model together with q. The budget counts memory entries in the form supplied by the backbone. We study this final selection step, which determines how the available evidence is assembled into the answer context.

The backbone handles memory writing, store organization, and candidate generation. To isolate the effect of memory selection, our controlled comparisons hold the store, candidate pool, budget, answerer, and evaluation protocol fixed.

2.2 MEMORY READING AS BUDGETED SET SELECTION

Let $U(q, S)$ denote the utility of memory set S for answering query q with a fixed answer model. Memory reading seeks a set that maximizes this utility within the available budget:

$$S^\star \in \arg\max_{\substack{S \subseteq \mathcal{C}_P(q) \\ |S| \leq K}} U(q, S). \tag{1}$$

Pointwise selection approximates set utility by adding individual relevance scores, $U_{\mathrm{pw}}(q, S) = \sum_{m \in S} g(q, m)$. For a fixed set size K, maximizing this sum gives the Top-K memories. Each memory contributes the same score regardless of what else is selected. The additive objective therefore cannot capture how overlapping evidence reduces a memory's contribution or how complementary facts become useful together. Since answer quality is not known at selection time, we need a computable objective that evaluates memories jointly. We use query reconstruction to measure how well a combination of memories represents the query and to guide the choice of that combination.

3 MEMPURSUIT

3.1 EVIDENCE COMPOSITION THROUGH SPARSE QUERY RECONSTRUCTION

To instantiate the set utility in Eq. 1, we evaluate how closely a weighted combination of memory embeddings can represent the query. This couples each memory's contribution to the rest of the set: similar embeddings offer similar reconstruction directions, while complementary embeddings can bring the joint reconstruction closer to the query.

Let $\mathcal{C} = \mathcal{C}_P(q)$, and let a fixed encoder $f : \mathcal{X} \to \mathbb{R}^d$ map query and memory texts to unit-norm embeddings. We formulate memory read as sparse query reconstruction over the full candidate pool:

$$\begin{aligned} \min_{\boldsymbol{\alpha} \in \mathbb{R}^P} \quad & \frac{1}{2} \left\| f(q) - \sum_{m \in \mathcal{C}} \alpha_m f(m) \right\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\alpha}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{\alpha} \succeq 0, \quad \sum_{m \in \mathcal{C}} \alpha_m = 1, \quad \|\boldsymbol{\alpha}\|_0 \leq K. \end{aligned} \tag{2}$$

The simplex constraints make the reconstruction a convex combination of selected embeddings. It prevents the fit from exploiting cancellation between positive and negative coefficients while fixing the total weight to prevent unrestricted rescaling. The cardinality constraint enforces the memory budget, and the ridge term with $\lambda > 0$ stabilizes the fit when memories overlap.

Eq. 2 defines a combinatorial optimization problem, but fitting the weights becomes convex once the memory set is fixed. We therefore use a fully corrective greedy pursuit that expands the set one memory at a time and jointly refits its weights (Tropp & Gilbert, 2007; Locatello et al., 2017). Each refit involves only the selected memories, and its residual guides the next expansion.

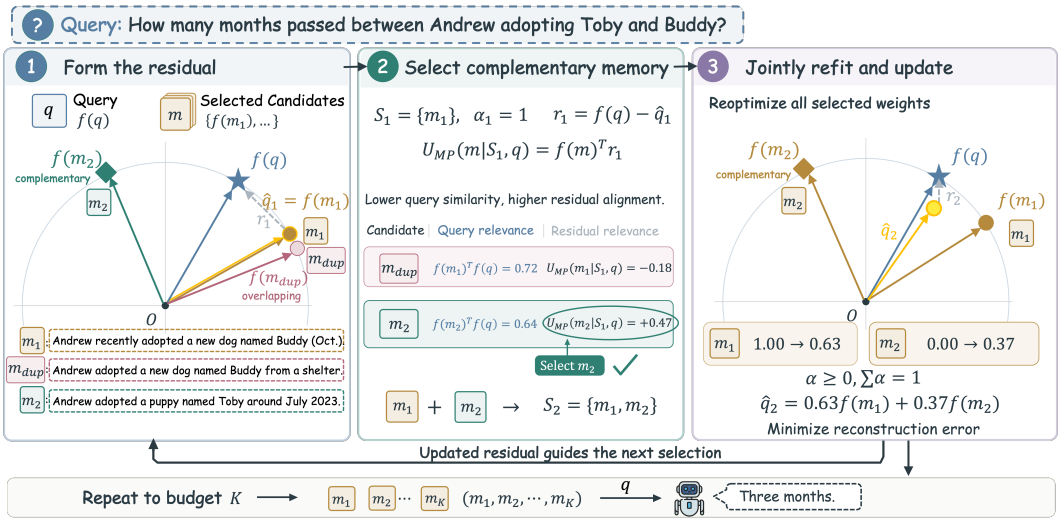


Figure 2: Overview of MEMPURSUIT, illustrated with a query requiring two adoption dates. (1) Form the residual between the query embedding and the current memory reconstruction. (2) Select the memory with the highest residual alignment: the complementary record about Toby is preferred over another record about Buddy, despite its lower query similarity. (3) Jointly refit the selected memories' nonnegative, sum-to-one weights and update the residual. Repeat until the memory budget is reached or the adaptive stopping condition is met. Scores and weights are illustrative.

3.2 FULLY CORRECTIVE RESIDUAL PURSUIT

The next memory should contribute to the part of the query that the current combination has not yet represented. For a selected set S with fitted weights $\alpha_m(q, S)$, this direction is captured by the reconstruction residual:

$$\hat{\mathbf{q}}(q, S) = \sum_{m \in S} \alpha_m(q, S) f(m), \qquad \mathbf{r}(q, S) = f(q) - \hat{\mathbf{q}}(q, S). \tag{3}$$

We evaluate an unselected memory by its alignment with this remaining direction:

$$U_{\mathrm{MP}}(m \mid q, S) = f(m)^\top \mathbf{r}(q, S). \tag{4}$$

Because an unselected memory has zero weight, this score equals the negative derivative of the objective in Eq. 2 with respect to its coefficient. Residual alignment thus provides a first-order criterion for expanding the selected set, without requiring a separate refit for every candidate. At step t, we select

$$m_t^\star = \arg \max_{m \in \mathcal{C} \setminus S_{t-1}} f(m)^\top \mathbf{r}(q, S_{t-1}), \qquad S_t = S_{t-1} \cup \{m_t^\star\}. \tag{5}$$

We initialize $S_0 = \varnothing$ and $\hat{\mathbf{q}}(q, S_0) = \mathbf{0}$, so the first selection is the strongest query cosine match. Subsequent selections depend on the residual left by the jointly fitted set.

Adding a memory can change how earlier selections should contribute to the reconstruction. Keeping their weights fixed would carry the previous allocation into the next selection, even when the expanded set admits a better fit. We therefore reoptimize all selected weights:

$$\boldsymbol{\alpha}_t = \underset{\substack{\boldsymbol{\alpha} \succeq 0 \\ \sum_{m \in S_t} \alpha_m = 1}}{\arg\min} \frac{1}{2} \left\| f(q) - \sum_{m \in S_t} \alpha_m f(m) \right\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\alpha}\|_2^2. \tag{6}$$

The fitted weights update the reconstruction and residual in Eq. 3. Selection identifies a direction for expanding the representation, while refitting redistributes contributions across the expanded set. Together, these operations make every subsequent choice depend on the current joint reconstruction.

3.3 CONTEXT CONSTRUCTION AND ADAPTIVE STOPPING

The greedy pursuit produces an ordered sequence of memories with nested selected sets $S_1 \subset S_2 \subset \cdots$. For a fixed budget K, MEMPURSUIT performs K selection–refit steps and uses the texts of the

first K selected memories as the answer context. A single pursuit path also supplies contexts for smaller budgets through its prefixes. Composition weights guide the search, while all selected entries remain in the context even if their fitted weights later become zero.

This incremental construction also allows context size to adapt to the query. Instead of always using the full budget, MEMPURSUIT can terminate when further selections make little progress in query reconstruction. After each refit, we measure the squared reconstruction error and its relative improvement:

$$L_t = \|\mathbf{r}(q, S_t)\|_2^2, \qquad \Delta_t = \frac{L_{t-1} - L_t}{\max(L_{t-1}, \epsilon_{\text{num}})}. \tag{7}$$

Here $\epsilon_{\text{num}} > 0$ stabilizes the denominator near zero. Adaptive selection stops when the relative improvement remains at or below a tolerance τ, or when the budget is reached. This yields a context of $k_q \leq K$ memories and avoids further context expansion and computation once reconstruction gains become small.

3.4 FROM RECONSTRUCTION TO EVIDENCE COVERAGE

The following proposition connects residual-guided selection to evidence coverage. It first bounds the scores of memories close to the selected set's convex hull. Under a fact-cluster model that groups memories supplying the same fact, it then gives conditions for selecting all required facts before repeating any of them.

Proposition 1 (Redundancy avoidance and evidence coverage)**.** *Assume unit-norm embeddings and exact refits with a fixed $\lambda \geq 0$. For nonempty S, let $H_S = \text{conv}\{f(s) : s \in S\}$. For $\delta \geq 0$, any candidate m with $\text{dist}(f(m), H_S) \leq \delta$ satisfies*

$$U_{\text{MP}}(m \mid q, S) \leq \max_{s \in S} f(s)^\top \mathbf{r}(q, S) + \delta \|\mathbf{r}(q, S)\|_2. \tag{8}$$

Thus, whenever an unselected memory exceeds this bound, the next selection lies outside the δ-neighborhood of H_S. Under the fact-cluster separation and coverage-margin conditions in Assumption B.1, MEMPURSUIT*, run for at least h steps, selects exactly one representative of each of the h required facts and no distractors in its first h selections. Consequently, every returned prefix with budget $K \geq h$ covers all required facts.*

The result explains how residual updates favor uncovered facts before spending additional slots on already represented evidence. Appendix B gives the full assumptions and proof, together with a computable optimality certificate for the reconstruction objective.

3.5 COMPLEXITY ANALYSIS

Table 1 compares the cost of final memory selection for MEMPURSUIT and a pointwise LLM scorer. Let P be the candidate count, d the embedding dimension, and $K \leq P$ the context budget. For the LLM scorer, L denotes the number of Transformer layers, W the hidden width, and n the maximum number of tokens per scoring sequence, including both prompt and continuation tokens. We analyze selection from available candidates and embeddings, excluding query encoding, candidate retrieval, and answer generation. Appendix A.2 details the complexity calculations.

Table 1: Time and space complexity of final memory selection. Numerical scales use $K = 50$, $P = 200$, $d = 1536$, $L = 32$, $W = 2560$, and $n = 128$.

	MEMPURSUIT		LLM scorer	
Resource	Asymptotic cost	Instantiated scale	Asymptotic cost	Instantiated scale
Time	$O(P^2d + KP^2 + K^4)$	6.969×10^7	$O(PL(nW^2 + n^2W))$	5.637×10^{12}
Space	$O(Pd + P^2)$	3.472×10^5	$O(LW^2 + LnW + n^2)$	2.202×10^8

MEMPURSUIT replaces repeated token-level neural evaluation with matrix operations on existing embeddings and small selected-set refits. This removes dependence on scorer depth and width while making each selection responsive to the evolving evidence context. Section 4.5 evaluates the resulting advantage in measured selection latency.

4 EXPERIMENTS

We compare MEMPURSUIT with nine baselines across three benchmarks using two answerers, and evaluate its integration into four memory frameworks. Further analyses examine evidence coverage, performance across question types and memory budgets, and quality–efficiency trade-offs with neural rerankers.

4.1 EXPERIMENTAL SETUP

Benchmarks & Baselines. We evaluate our proposed method end-to-end on 3 well-known benchmarks: LoCoMo (Maharana et al., 2024), LongMemEval (Wu et al., 2025), and BEAM (Tavakoli et al., 2026), with different average context lengths of 16.6k, 115k, and 500k tokens respectively. The system comparison includes full context, Pure RAG, A-MEM (Xu et al., 2025), LightMem (Fang et al., 2026), SimpleMem (Liu et al., 2026), MemoryOS (Kang et al., 2025), Mem0 (Chhikara et al., 2025), MemoryBank (Zhong et al., 2024), and MemOS (Li et al., 2025).

Evaluation protocol. In the end-to-end experiments, we evaluate MEMPURSUIT and baselines with two answer LLMs, GPT-4o-mini (OpenAI, 2024) and Qwen3-30B-A3B-Instruct-2507 (Qwen Team, 2025). All systems use the same answer prompt, judge model, and judge prompt. Full-system comparisons retain each baseline's native memory construction and retrieval settings. The MEMPURSUIT pipeline in Table 2 uses Mem0-v3 OSS as its memory backbone with a memory read budget of $K = 50$. We report binary-judge QA accuracy for LoCoMo and LongMemEval-S and the official grader score for BEAM.

To evaluate whether MEMPURSUIT improves memory reading across different frameworks, we compare LightMem, MemoryBank, SimpleMem, and Mem0 with and without MEMPURSUIT on LoCoMo at $K \in \{10, 20\}$. For each framework, the two variants share the same memory store, candidate pool, and selection budget, with GPT-4o-mini serving as both answerer and judge. We fix the simplex-refit ridge coefficient to $\lambda = 10^{-10}$ across all experiments. Full implementation details are provided in Appendix A.

4.2 END-TO-END PERFORMANCE AND CROSS-FRAMEWORK COMPARISONS

Strong performance across long-term memory benchmarks. MEMPURSUIT with Mem0-v3 backbone achieves the highest aggregate score among the evaluated systems on all three benchmarks with both answerers as demonstrated in Table 2. With GPT-4o-mini, it reaches 87.08% on LoCoMo, 71.20% on LongMemEval-S, and 43.82 on BEAM. The overall lead persists with Qwen3-30B-A3B, demonstrating effective memory reading across downstream models and interaction histories.

Reusable reading improvement when context is scarce. The cross-framework comparisons in Table 3 show higher overall accuracy in all eight framework–budget configurations with MEMPURSUIT. At Top-10, gains range from 2.47% to 9.29%; improvements also hold at Top-20. The largest and most consistent gains occur on multi-hop and single-hop questions, with accuracy improvements ranging from 0.71% to 10.28% and from 0.59% to 10.11% respectively across the eight configurations. These results show that MEMPURSUIT improves memory reading across frameworks with different memory construction strategies, enabling each framework to make more effective use of a limited context budget. The following analysis traces these gains to the evidence entering the selected context.

4.3 DOES MEMPURSUIT RECALL WHAT IS MISSING?

To examine whether MemPursuit recalls evidence missed by relevance-based selection, we track how it changes the selected context and its coverage of annotated gold sessions. We compare MemPursuit with the original hybrid order on LoCoMo at Top-10 and Top-20, using the same pool of 200 candidates per question and keeping the rest of the Mem0-v3 pipeline fixed in Figure 3. MEMPURSUIT replaces 70.8% of the baseline's selected memories at Top-10 and 64.5% at Top-20. These changes increase macro gold-session recall from 82.9% to 90.3% at Top-10 and from 91.3% to 94.8% at Top-20. Complete coverage rises from 77.1% to 84.5% and from 86.2% to 90.3%, respectively. Newly recalled memories contribute more gold-session coverage than is lost through displaced memories. Residual-guided selection therefore brings previously omitted evidence sources into the context, allowing the same memory budget to cover more of the query-required sources.

Table 2: End-to-end results under two answerer backbones. LoCoMo and LongMemEval-S report binary-judge accuracy (%); BEAM reports its official mean score scaled to 0–100. Bold and underline denote the best and second-best result within each answerer panel. "–" denotes an unavailable run due to limited context length.

	LoCoMo					LongMemEval-S							BEAM
Method	MH	Temp.	Open	SH	Avg.	KU	MS	SSA	SSP	SSU	TR	Avg.	Avg.
GPT-4o-mini													
Full context	68.79	50.16	56.25	86.56	73.83	–	–	–	–	–	–	–	–
RAG	82.62	58.57	64.58	85.85	78.25	53.85	45.86	**94.64**	46.67	87.14	41.35	57.20	38.23
A-MEM	59.22	64.17	54.17	69.80	65.71	69.23	55.64	**94.64**	40.00	84.29	63.16	67.20	38.78
LightMem	79.08	73.83	62.50	86.56	81.04	**80.77**	72.93	12.50	46.67	88.57	**77.44**	69.20	41.42
SimpleMem	79.79	53.27	64.58	86.21	76.82	53.85	38.35	33.93	40.00	71.43	41.35	45.80	36.37
MemoryOS	53.90	46.42	42.71	63.26	56.75	70.51	57.89	75.00	43.33	74.29	55.64	62.60	31.05
Mem0	64.54	53.89	51.04	72.41	65.78	78.21	**78.20**	16.07	46.67	**92.86**	74.44	70.40	36.05
MemoryBank	32.27	23.68	44.79	47.80	39.74	39.74	20.30	85.71	36.67	70.00	26.32	40.20	34.67
MemOS	82.98	77.57	63.54	90.01	84.48	71.79	64.66	35.71	50.00	82.86	69.17	65.40	35.77
MEMPURSUIT	**84.40**	**80.06**	**65.62**	**93.10**	**87.08**	79.49	73.68	57.14	**60.00**	**92.86**	60.90	**71.20**	**43.82**
Qwen3-30B-A3B-Instruct-2507													
Full context	69.86	51.71	57.29	87.40	74.87	–	–	–	–	–	–	–	–
RAG	81.21	72.59	66.67	90.73	83.70	67.95	44.36	92.86	53.33	90.00	46.62	61.00	35.22
A-MEM	58.16	61.06	61.46	69.32	65.06	73.08	54.14	**96.43**	83.33	94.29	63.91	71.80	34.27
LightMem	78.37	72.27	63.54	84.66	79.61	88.46	**72.18**	16.07	**86.67**	91.43	72.93	72.20	38.88
SimpleMem	78.01	54.83	**70.83**	84.54	76.30	56.41	39.10	44.64	70.00	82.86	42.86	51.40	37.39
MemoryOS	56.74	47.35	52.08	64.21	58.57	69.23	57.14	73.21	80.00	82.86	66.92	68.40	31.12
Mem0	64.18	53.27	55.21	68.97	63.96	83.33	69.92	17.86	73.33	**97.14**	**75.94**	71.80	37.28
MemoryBank	36.17	22.74	51.04	49.11	41.36	42.31	20.30	89.29	66.67	75.71	31.58	45.00	32.77
MemOS	76.24	78.19	68.75	85.85	81.43	78.21	66.17	41.07	76.67	88.57	74.44	71.20	36.08
MEMPURSUIT	**81.91**	**80.37**	63.54	**90.96**	**85.39**	**91.03**	71.43	62.50	76.67	**97.14**	62.41	**75.00**	**43.66**

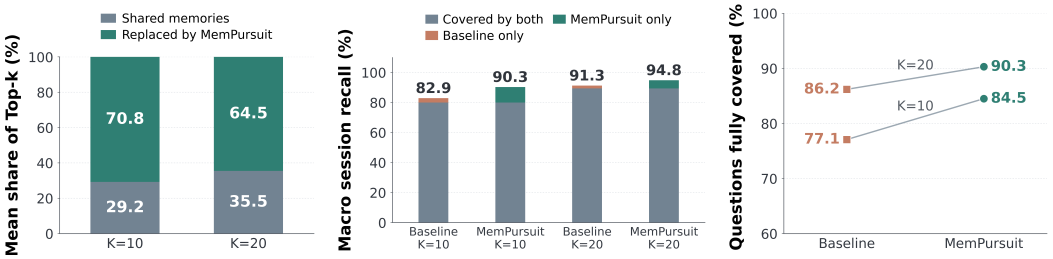
Abbreviations. MH = multi-hop, Temp. = temporal, Open = open-domain, SH = single-hop. KU = knowledge update, MS = multi-session, SSA = single-session assistant, SSP = single-session preference, SSU = single-session user, TR = temporal reasoning.

Table 3: Cross-framework comparisons on LoCoMo at $K = 10$. Each system is evaluated without (w/o) and with (w/) MEMPURSUIT at the same selection budget. Entries report accuracy (%); parentheses in the w/ rows give the change in percentage points from the paired w/o result. For each system, the higher accuracy in each metric is bolded. Yellow highlights the w/ results. All configurations evaluate 1,540 questions using GPT-4o-mini as answerer and judge. Full results at $K \in \{10, 20\}$ appear in Table 4 (Appendix C.1).

System	MEMPURSUIT	Multi-hop	Temporal	Open-domain	Single-hop	Overall
LightMem	w/o	62.77	66.98	56.25	77.17	71.10
	w/	**69.15** (+6.38)	**68.22** (+1.25)	**57.29** (+1.04)	**79.19** (+2.02)	**73.70** (+2.60)
MemoryBank	w/o	53.55	**43.30**	51.04	65.87	57.99
	w/	**56.74** (+3.19)	39.25 (−4.05)	**56.25** (+5.21)	**70.27** (+4.40)	**60.45** (+2.47)
SimpleMem	w/o	62.41	48.60	56.25	74.91	65.97
	w/	**70.21** (+7.80)	**49.22** (+0.62)	**60.42** (+4.17)	**82.28** (+7.37)	**71.82** (+5.84)
Mem0	w/o	45.04	42.37	43.75	57.43	51.17
	w/	**55.32** (+10.28)	**49.84** (+7.48)	**48.96** (+5.21)	**67.54** (+10.11)	**60.45** (+9.29)

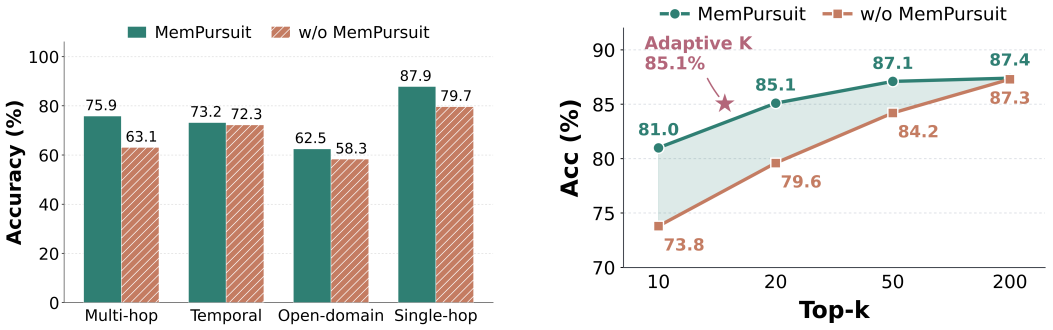
4.4 PERFORMANCE ACROSS QUESTION TYPES AND CONTEXT BUDGETS

We next examine how the changes in selected evidence translate into answer accuracy across question types and memory budgets. Using the same candidate pools and evaluation setup as Section 4.3, we compare MEMPURSUIT with the original hybrid order.



(a) Memory selection overlap (b) Gold evidence-session coverage (c) Complete evidence-session coverage

Figure 3: Memory selection and evidence-session coverage on LoCoMo at Top-10 and Top-20. Panels (b) and (c) evaluate 1,536 gold-annotated questions. MEMPURSUIT changes the selected context while increasing both average and complete evidence-session coverage.



(a) Question-type accuracy at Top-10. (b) Overall accuracy across Top-K context budgets.

Figure 4: LoCoMo performance with GPT-4o-mini as the answerer. The star in panel (b) marks adaptive context selection using the stopping rule in Section 3.3 with $\tau = 10^{-5}$, achieving 85.1% accuracy with an average of 15.65 selected memories. The 0.1% accuracy gap at Top-200 arises from differences in memory ordering.

The strongest gain occurs on multi-hop questions. In Figure 4a, Multi-hop accuracy rises from 63.1% to 75.9%, while single-hop accuracy increases from 79.7% to 87.9% . Temporal and open-domain questions also improve at this budget. The particularly large multi-hop gain fits the evidence-composition mechanism: allocating context across complementary sources is valuable when several facts must be available together. Single-hop gains show that more effective context allocation also benefits focused factual recall.

Better evidence enters the context earlier. In Figure 4b, MEMPURSUIT reaches 85.1% accuracy with 20 memories, exceeding hybrid retrieval with 50 memories. It can be observed that the advantage is largest at the smallest tested budget and narrows as the two policies include more of the candidate pool. Moreover, MEMPURSUIT already approaches its Top-200 accuracy with only 50 memories. These patterns show that MEMPURSUIT assembles useful evidence within a smaller memory budget, whereas relevance-based ordering requires a larger context to achieve comparable answer accuracy.

4.5 QUALITY–EFFICIENCY TRADE-OFFS WITH NEURAL RERANKERS

Finally, we compare MEMPURSUIT with BGE-base, BGE-large, and BGE-v2-m3 cross-encoder rerankers (Xiao et al., 2023; Chen et al., 2024), as well as Qwen3.5-0.8B, -2B, and -4B pointwise LLM scorers (Qwen Team, 2026) in Figure 5. Appendix C.7 provides the complete latency statistics and batch-size comparisons. Relative to hybrid retrieval, MEMPURSUIT captures approximately 73% of the accuracy gain delivered by BGE-v2-m3, the most accurate reranker in this comparison, while requiring only 3.2% of its median reranking latency. It also exceeds BGE-base and Qwen3.5-0.8B in answer accuracy and comes within half a percentage point of BGE-large and Qwen3.5-2B. Even when BGE-v2-m3 processes all 200 candidates in a single batch, MEMPURSUIT achieves a 32.9× speedup in median reranking latency. MEMPURSUIT therefore combines strong answer quality with low online selection latency, even against fully batched neural scoring.

4.6 Additional experimental results

Additional experiments test whether gold evidence reconstructs the query more closely than retrieved alternatives (Appendix C.2) and whether residual scoring favors missing evidence given an already selected evidence memory (Appendix C.3). Case studies are in the same appendix. We compare residual-guided selection with MMR in Appendix C.5. Appendix C.7 reports latency comparisons with neural rerankers across batch sizes and examines how MemPursuit's selection latency scales with candidate count P and budget K.

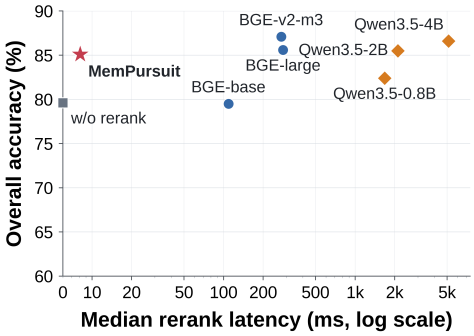


Figure 5: LoCoMo accuracy versus median reranking latency at Top-20 with $P = 200$. Neural scorers use batch size 128. Latency includes final reranking only.

5 Related Work

Long-term memory for language-model agents. Agent memory systems improve retention and access through persistent memory management (Zhong et al., 2024; Chhikara et al., 2025), segmentation and compression (Pan et al., 2025; Fang et al., 2026; Liu et al., 2026), and linked or graph-structured organization (Xu et al., 2025; Gutiérrez et al., 2024; Rasmussen et al., 2025). MemoryOS and MemOS develop broader memory-management architectures (Kang et al., 2025; Li et al., 2025), while Hindsight distinguishes factual evidence, experiences, and beliefs (Latimer et al., 2025). Memory-R1 learns memory management and utilization through reinforcement learning (Yan et al., 2025b). These systems shape what agents retain and how they access it. MemPursuit focuses on composing the final answer context from a supplied candidate pool, providing a reusable read operation across memory architectures.

From relevance ranking to evidence composition. Dense retrieval such as DPR (Karpukhin et al., 2020) and pointwise rerankers such as MemReranker (Li et al., 2026) estimate individual relevance, whereas RankGPT and ListT5 use listwise comparisons to produce rankings (Sun et al., 2023; Yoon et al., 2024). Set-dependent selection has a longer history in MMR (Carbonell & Goldstein, 1998), xQuAD (Santos et al., 2010), submodular optimization (Lin & Bilmes, 2011), and determinantal point processes (Kulesza & Taskar, 2012), which promote novelty, aspect coverage, or diversity. More recent approaches target evidence utility directly (Deng et al., 2025; Singh & Vadlamani, 2026). For instance, SetR uses LLM reasoning to select passages satisfying query requirements (Lee et al., 2025). MemPursuit instead evaluates the selected set through query reconstruction. Jointly refitted weights determine the residual guiding each addition, making selection sensitive to what the current set already represents without a separately trained utility model or LLM-based selection.

Dynamic retrieval and residual pursuit. MemR^3 and GAM interleave retrieval with reflection or research (Du et al., 2025; Yan et al., 2025a), while MRAgent uses LLM reasoning to prune memory-graph paths (Ji et al., 2026). xMemory combines hierarchical organization with uncertainty-guided expansion (Hu et al., 2026). These approaches make retrieval responsive to accumulated evidence. MemPursuit implements this adaptation geometrically within the candidate pool. Drawing on orthogonal matching pursuit (Tropp & Gilbert, 2007), Frank–Wolfe optimization (Jaggi, 2013), and generalized matching pursuit (Locatello et al., 2017), it alternates residual-guided selection with fully corrective simplex refitting. This connects sparse approximation to agent-memory composition, yielding nested contexts through inexpensive operations on existing embeddings.

6 Conclusion

A long-term memory system must compose a limited answer context from complementary evidence. MemPursuit implements this operation through simplex-constrained residual pursuit: selected memories jointly reconstruct the query, and the remaining residual guides subsequent recall. The method is training-free, reuses existing embeddings, and produces nested contexts without additional neural scoring. Experiments show consistent accuracy gains across memory frameworks, while comparisons with neural rerankers demonstrate competitive answer quality at a fraction of their selection cost. However, MemPursuit relies on the structure of the semantic embedding space and cannot explicitly reason about contextual relationships between memories and the query.

AI USE STATEMENT

We used generative AI tools to assist with polishing the manuscript, improving its organization and clarity, and identifying and checking relevant references. These tools were not used to research topic selection or the methodology design. All AI-assisted content was reviewed and revised by the authors, and technical statements and references were checked before inclusion. The authors take full responsibility for the final manuscript and its scientific content.

REPRODUCIBILITY STATEMENT

To support reproducibility, we document key experimental settings and parameters in Appendix A, including memory construction and retrieval, selection budgets, MEMPURSUIT hyperparameters, and answer-generation and evaluation prompts. Settings for additional experiments accompany their results in Appendix C, with hardware configurations and execution details reported in Appendix C.7. The core implementation code is included in the supplementary material.

REFERENCES

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*, SIGIR98, pp. 335–336. ACM, August 1998. doi: 10.1145/290941.291025. URL http://dx.doi.org/10.1145/290941.291025.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. BGE M3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024. URL https://arxiv.org/abs/2402.03216.

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.

Jiale Deng, Yanyan Shen, Ziyuan Pei, Youmin Chen, and Linpeng Huang. Influence Guided Context Selection for Effective Retrieval-Augmented Generation. *arXiv preprint arXiv:2509.21359*, 2025. doi: 10.48550/ARXIV.2509.21359. URL https://arxiv.org/abs/2509.21359.

Xingbo Du, Loka Li, Duzhen Zhang, and Le Song. MemR^3: Memory Retrieval via Reflective Reasoning for LLM Agents. *arXiv preprint arXiv:2512.20237*, 2025. doi: 10.48550/ARXIV.2512.20237. URL https://arxiv.org/abs/2512.20237.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dyJ0GWpjJB.

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*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/6ddc001d07ca4f319af96a3024f6dbd1-Abstract-Conference.html.

Zhanghao Hu, Qinglin Zhu, Runcong Zhao, Di Liang, Hanqi Yan, Yulan He, and Lin Gui. Beyond RAG for Agent Memory: Retrieval by Decoupling and Aggregation. *arXiv preprint arXiv:2602.02007*, 2026. doi: 10.48550/ARXIV.2602.02007. URL https://arxiv.org/abs/2602.02007.

Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 427–435. PMLR, 2013. URL https://proceedings.mlr.press/v28/jaggi13.html.

Shuo Ji, Yibo Li, and Bryan Hooi. Memory is Reconstructed, Not Retrieved: Graph Memory for LLM Agents. *arXiv preprint arXiv:2606.06036*, 2026. doi: 10.48550/ARXIV.2606.06036. URL https://arxiv.org/abs/2606.06036.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. *arXiv preprint arXiv:2506.06326*, 2025. URL https://arxiv.org/abs/2506.06326.

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*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Alex Kulesza and Ben Taskar. Determinantal Point Processes for Machine Learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286, December 2012. ISSN 1935-8245. doi: 10.1561/2200000044. URL http://dx.doi.org/10.1561/2200000044.

Chris Latimer, Nicoló Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight is 20/20: Building Agent Memory that Retains, Recalls, and Reflects. *arXiv preprint arXiv:2512.12818*, 2025. doi: 10.48550/ARXIV.2512.12818. URL https://arxiv.org/abs/2512.12818.

Dahyun Lee, Yongrae Jo, Haeju Park, and Moontae Lee. Shifting from ranking to set selection for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17606–17619. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.861. URL https://aclanthology.org/2025.acl-long.861/.

Chunyu Li, Jingyi Kang, Ding Chen, Mengyuan Zhang, Jiajun Shen, Bo Tang, Xuanhe Zhou, Feiyu Xiong, and Zhiyu Li. MemReranker: Reasoning-aware reranking for agent memory retrieval. *arXiv preprint arXiv:2605.06132*, 2026. URL https://arxiv.org/abs/2605.06132.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, Junpeng Ren, Zehao Lin, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhiqiang Yin, Qingchen Yu, Bo Tang, Hongkang Yang, Zhi-Qin John Xu, and Feiyu Xiong. MemOS: An operating system for memory-augmented generation in large language models. *arXiv preprint arXiv:2505.22101*, 2025. URL https://arxiv.org/abs/2505.22101.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 510–520, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1052/.

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

Francesco Locatello, Rajiv Khanna, Michael Tschannen, and Martin Jaggi. A unified optimization view on generalized matching pursuit and Frank–Wolfe. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 860–868. PMLR, 2017. URL https://proceedings.mlr.press/v54/locatello17a.html.

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. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://aclanthology.org/2024.acl-long.747/.

OpenAI. GPT-4o mini: Advancing cost-efficient intelligence, July 2024. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xKDZAW0He3.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

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

Rodrygo L.T. Santos, Craig Macdonald, and Iadh Ounis. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pp. 881–890. ACM, April 2010. doi: 10.1145/1772690.1772780. URL http://dx.doi.org/10.1145/1772690.1772780.

Rahul Singh and Madhav Vadlamani. QUBO-Optimized Evidence Selection for Retrieval-Augmented Question Answering with Unconventional Solvers. *arXiv preprint arXiv:2607.12334*, 2026. doi: 10.48550/ARXIV.2607.12334. URL https://arxiv.org/abs/2607.12334.

Jiashuo Sun, Pengcheng Jiang, Saizhuo Wang, Jiajun Fan, Heng Wang, Siru Ouyang, Ming Zhong, Yizhu Jiao, Chengsong Huang, Xueqiang Xu, Pengrui Han, Peiran Li, Jiaxin Huang, Ge Liu, Heng Ji, and Jiawei Han. Rethinking the Reranker: Boundary-Aware Evidence Selection for Robust Retrieval-Augmented Generation. *arXiv preprint arXiv:2602.03689*, 2026. doi: 10.48550/ARXIV.2602.03689. URL https://arxiv.org/abs/2602.03689.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14918–14937. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.923. URL https://aclanthology.org/2023.emnlp-main.923/.

Mohammad Tavakoli, Alireza Salemi, Carrie Ye, Mohamed Abdalla, Hamed Zamani, and J. Ross Mitchell. Beyond a million tokens: Benchmarking and enhancing long-term memory in LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=y59hf5lrMn.

Joel A. Tropp and Anna C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, December 2007. ISSN 0018-9448. doi: 10.1109/tit.2007.909108. URL http://dx.doi.org/10.1109/TIT.2007.909108.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-Pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*, 2023. URL https://arxiv.org/abs/2309.07597.

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, Main Conference, pp. 17577–17604. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/19909c36f51abc4856b4560aff3d36d6-Paper-Conference.pdf.

B. Y. Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General Agentic Memory Via Deep Research. *arXiv preprint arXiv:2511.18423*, 2025a. doi: 10.48550/ARXIV.2511.18423. URL https://arxiv.org/abs/2511.18423.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing Large Language Model Agents to Manage and Utilize Memories via Reinforcement Learning. *arXiv preprint arXiv:2508.19828*, 2025b. doi: 10.48550/ARXIV.2508.19828. URL https://arxiv.org/abs/2508.19828.

Soyoung Yoon, Eunbi Choi, Jiyeon Kim, Hyeongu Yun, Yireun Kim, and Seung-won Hwang. ListT5: Listwise reranking with fusion-in-decoder improves zero-shot retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2287–2308. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.125. URL https://aclanthology.org/2024.acl-long.125/.

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*, volume 38, pp. 19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946. URL https://ojs.aaai.org/index.php/AAAI/article/view/29946.

Wei Zhou, Xuanhe Zhou, Shaokun Han, Hongming Xu, Guoliang Li, Zhiyu Li, Feiyu Xiong, and Fan Wu. Are we ready for an agent-native memory system? *arXiv preprint arXiv:2606.24775*, 2026. URL https://arxiv.org/abs/2606.24775.

A Implementation Details

A.1 MemPursuit algorithm and implementation

Input: query text q; candidate memories $\mathcal{C} = \mathcal{C}_P(q)$; query and memory embeddings $f(q), \{f(m) : m \in \mathcal{C}\}$ from the fixed normalized encoder f; memory budget K; ridge λ.
Output: memories in residual-guided entry order.

1. Index $\mathcal{C} = \{m_1, \ldots, m_P\}$; compute $G_{ij} = f(m_i)^\top f(m_j)$ and $b_i(q) = f(m_i)^\top f(q)$; initialize $S_0 \leftarrow \varnothing$.
2. For $t = 1, \ldots, K$:
   (a) Compute $U_{\mathrm{MP}}(m_i \mid q, S_{t-1}) \leftarrow b_i(q) - \sum_{m_j \in S_{t-1}} G_{ij}\alpha_{m_j}(q, S_{t-1})$ for all candidates (use $U_{\mathrm{MP}}(m_i \mid q, S_0) = b_i(q)$) and mask selected memories.
   (b) Select $m_t^\star \leftarrow \arg\max_{m \in \mathcal{C} \setminus S_{t-1}} U_{\mathrm{MP}}(m \mid q, S_{t-1})$ and set $S_t \leftarrow S_{t-1} \cup \{m_t^\star\}$.
   (c) Solve Eq. 6 for $\boldsymbol{\alpha}_t$ on S_t, warm-started from the previous weights and a zero weight for $m_t^\star$.
   (d) Record residual norm, marginal objective reduction, and composition weights.
3. Return $(m_1^\star, \ldots, m_K^\star)$; its first k memories form $S_k(q)$ for any $k \leq K$.

Figure 6: Fixed-budget MemPursuit procedure. Memories are returned in entry order, without sorting by final coefficient magnitude.

A.1.1 Numerical implementation

Precomputed similarities. Index the candidate memories as $\mathcal{C} = \{m_1, \ldots, m_P\}$ and precompute the memory–memory Gram matrix $G_{ij} = f(m_i)^\top f(m_j)$ and query–memory similarities $b_i(q) = f(m_i)^\top f(q)$. The refit vector $\boldsymbol{\alpha}_t$ contains the weights $\alpha_m(q, S_t)$ for $m \in S_t$. Using Eq. 4, the conditional selection score for memory m_i is

$$\begin{aligned} U_{\mathrm{MP}}(m_i \mid q, S_{t-1}) &= b_i(q) - \sum_{m_j \in S_{t-1}} G_{ij}\alpha_{m_j}(q, S_{t-1}) \\ &= f(m_i)^\top \mathbf{r}(q, S_{t-1}), \end{aligned} \tag{9}$$

allowing all candidates to be evaluated against the updated reconstruction using the precomputed Gram matrix and query similarities. These quantities are reused throughout the pursuit path.

Warm starting the refit. For $t > 1$, each refit starts from the previous solution with a zero coefficient appended for the new memory. This initialization preserves simplex feasibility and reuses the preceding computation. All selected coefficients are then jointly optimized according to Eq. 6.

Selection and output semantics. With a fixed budget, MemPursuit runs for K selections and returns $(m_1^\star, \ldots, m_K^\star)$ in greedy entry order. For any $k \leq K$, the first k entries form the evidence set $S_k(q) = \{m_1^\star, \ldots, m_k^\star\}$. The query argument in $S_t(q)$ is omitted in the main derivation for readability. Composition weights guide subsequent selection; a memory remains in the path even if its weight later becomes zero. The returned path is not reordered by final coefficient magnitude.

Adaptive stopping. Adaptive stopping applies Eq. 7 after each refit, with K as the maximum number of selected memories. The returned context includes all entries selected up to termination. For the adaptive-budget experiment in Section 4.4, we set the numerical floor $\epsilon_{\mathrm{num}} = 10^{-12}$, the relative-improvement threshold $\tau = 10^{-5}$, and the patience $T_{\mathrm{pat}} = 2$. Thus selection stops after two consecutive iterations with relative reconstruction-error improvement at most 10^{-5}. The stopping loss is the squared residual norm L_t in Eq. 7, excluding the ridge penalty.

A.2 Computational complexity

Let P be the candidate count, d the embedding dimension, and $K \leq P$ the context budget. We bound the cost of up to K pursuit steps. We analyze selection from available candidates and embeddings, excluding query encoding, candidate retrieval, and answer generation.

MemPursuit. The full-Gram implementation constructs the Gram matrix in $O(P^2d)$ time. At step t, candidate scoring costs $O(Pt)$, and a dense full-candidate gradient refresh costs $O(P^2)$. For the t-variable simplex QP, an initial factorization costs $O(t^3)$; updating it across $O(t)$ working-set iterations at $O(t^2)$ per update gives $O(t^3)$ per refit. With one refit per step and at most K steps, the total time is $O(P^2d + PK^2 + KP^2 + K^4)$, which simplifies to $O(P^2d + KP^2 + K^4)$ because $K \leq P$. The candidate embeddings, full Gram matrix, and QP factors occupy $O(Pd + P^2 + K^2) = O(Pd + P^2)$ working memory. A single pursuit path supplies all smaller context budgets. Adaptive stopping can reduce the number of steps and the associated selection and refitting costs.

LLM-based scoring complexity. For a dense Transformer with L layers and hidden width W, let n bound the total number of prompt and continuation tokens in a scoring sequence. Linear projections and feed-forward transformations cost $O(nW^2)$ per layer, while attention costs $O(n^2W)$ (Vaswani et al., 2017). Scoring P query–memory pairs therefore costs

$$T_{\mathrm{LLM}} = O\big(PL(nW^2 + n^2W)\big)\,. \tag{10}$$

For single-sequence inference, model parameters require $O(LW^2)$ space, the key–value cache requires $O(LnW)$, and materialized attention requires $O(n^2)$ working space. The resulting space complexity is

$$S_{\mathrm{LLM}} = O(LW^2 + LnW + n^2). \tag{11}$$

These are the time and space estimates reported in Table 1; the space estimate includes scorer parameters and inference states for one sequence.

A.3 Memory backbones and baseline configurations

Mem0-v3 memory construction and retrieval. The Mem0-v3 implementation processes timestamped conversations incrementally. For each incoming message batch, GPT-4o-mini at temperature 0.1 extracts new, self-contained memories using the current messages, the 10 most recent messages, and 10 relevant existing memories retrieved by dense similarity. Facts can be extracted from both user and assistant messages, with observation timestamps resolving relative time expressions. Extraction appends new memories without overwriting earlier records. Exact duplicates are filtered against the retrieved memories and within the current batch. The resulting memories are stored in Qdrant with `text-embedding-3-small` embeddings ($d = 1536$), BM25 sparse vectors derived from lemmatized text, and temporal and source metadata. Extracted entities form a separate entity-to-memory index.

At query time, dense and BM25 retrieval produce a union of candidates, which are ranked by combining dense similarity, calibrated BM25 scores, and an entity-matching boost. Entity matching adjusts scores within this union without adding candidates. The top $P = 200$ memories form the candidate pool supplied to MemPursuit. After MemPursuit selects the requested prefix of its pursuit path, the selected memories are sorted by timestamp for presentation to the answer model. This step preserves the selected set while presenting events and updates in chronological order.

System-comparison retrieval budgets. We follow the baselines' native retrieval configurations, with an adjusted budget for MemoryBank. A-MEM and MemoryOS use $K = 10$. A-MEM additionally includes linked memories. MemoryOS supplements retrieved memories with short-term memory, user and assistant knowledge, and a user profile. LightMem uses $K = 60$ across LoCoMo, LongMemEval-S, and BEAM. SimpleMem includes all stored memories, while Mem0 uses $K = 60$. For MemoryBank, we increase its default budget from $K = 2$ to $K = 50$ to match MemPursuit's nominal retrieval budget. MemOS retrieves the top 20 memories separately for each of the two mirrored speakers, yielding 40 memory entries in the answer context.

Cross-framework comparisons. Paired variants share the stored memories, candidate pool, and downstream processing. We reuse each framework's query and memory embeddings, apply ℓ_2 normalization, and run MemPursuit with early stopping disabled for up to 200 selections, limited by the available candidates. Selections at $K = 10$ and $K = 20$ are prefixes of the same greedy selection sequence. Selection follows pursuit order rather than the fitted coefficients, without blending in the original retrieval scores. The Mem0-v3 hybrid pool is described above. The four frameworks in Tables 3 and 4 are integrated as follows:

- **LightMem.** Both variants reuse the saved global Top-60 retrieval pool. The baseline retains the native retrieval order, while MEMPURSUIT selects from the same pool.
- **SimpleMem.** We retain the complete deduplicated union of semantic, keyword, and structured retrieval results, using the embedding vectors from LanceDB. Both variants truncate this same pool at K, following either the original hybrid order or the pursuit order.
- **MemoryBank.** We retrieve the Top-200 seeds by native L2 distance, then select K seeds by either L2 rank or MEMPURSUIT. Both variants subsequently apply the same within-date neighbor expansion, merging, chronological ordering, and resident summaries. Thus, K counts seeds, and the final number of context chunks can differ from K. These $K \in \{10, 20\}$ comparisons are separate from the $K = 50$ system-comparison baseline.
- **Mem0 Paper OSS.** We merge the two mirrored speaker stores' Top-30 results into a global pool of at most 60 memories. The baseline orders this union by saved native cosine scores, while MEMPURSUIT runs once over the combined pool before truncation at K.

A.4 BENCHMARK PROMPT TEMPLATES

We reproduce the LoCoMo prompt templates below and specify the prompt sources for LongMemEval and BEAM.

A.4.1 LOCOMO

The answer and judge template comes from the implementation of Mem0 (Chhikara et al., 2025). Its placeholders are `{question}`, `{reference_date}`, and `{memories}`. The builder defaults the reference date to `2023`, takes up to 200 supplied memories, and presents them oldest first with human-readable dates. An optional user profile is prepended to the memories field. No retrieval scores or ranks are shown.

LoCoMo — Answer generation

You are answering a question using retrieved memories from past conversations. Follow these reasoning steps IN ORDER.

Step 1: SCAN ALL MEMORIES
Read EVERY memory below from first to last. For each one that contains information relevant to the question, note it. Do NOT stop after finding the first relevant memory — important details are often scattered across many memories, including ones far down the list. Give equal weight to ALL memories regardless of position — a memory near the end is just as likely to contain the answer as one near the beginning. In these memories, "User" refers to the main person whose memories these are.

Step 2: ENTITY VERIFICATION
Confirm each relevant memory is about the correct person/entity. If the question asks "What does Person A like?" and a memory says "Person B likes X", do NOT use that memory to answer about Person A. In two-person conversations, both speakers' actions are relevant — if the question asks about person A and a memory attributes an action to person B (the other speaker), that information is still valid evidence from their shared conversations, but always check the attribution is correct.

Step 3: COMBINE AND CROSS-REFERENCE
- COMBINE facts from multiple memories about the same topic. If one memory says "won first place" and another says "performed a piece titled X," those describe the same event — connect them.
- For listing/counting questions, extract EVERY distinct item from ALL memories. A single memory may contain multiple items. Think about what CATEGORIES of answers the question could have, then re-scan specifically for each category.
- For counting questions ("how many times", "how many X"), enumerate each distinct instance explicitly with its date or context BEFORE giving a final count. Do not estimate — list them out, then count the list.
- DECOMPOSE complex sentences: "an immersive X with Y, enjoys Z" contains multiple distinct facts. Each could be the answer.
- Connect related facts across memories: if one says "nearby lake" and another says "Lake Tahoe is great for kayaking", the nearby lake IS Lake Tahoe. If one says "bought X in Paris", infer the country is France.

Step 4: SELECT THE BEST ANSWER
- Do NOT assume the highest-ranked memory is correct. Multiple memories may describe different events for the same topic. Compare each candidate's relevance to the SPECIFIC question, not its retrieval score. A lower-ranked memory that directly answers the question beats a higher-ranked one that is only tangentially related.
- ALWAYS choose the MOST SPECIFIC detail available. A proper name, title, or number beats a generic description. Rate each candidate as HIGH specificity (name, title, number, specific activity) or LOW (generic description), and prefer HIGH.

LoCoMo — Answer generation (continued)

- Report what someone actually DID, not what was offered or available to them. "Has not tried X yet" means X was NOT done — disqualify it. "Joined X" or "has done X" means it WAS done — prefer it.
- When multiple memories repeat the same generic fact, that repetition does NOT make it more correct than a single memory with a more specific answer.
- Photos depict what was IN the photo, not facts about someone's daily life. Prefer direct statements over photo descriptions for inferences.
- Re-read the question carefully before answering. If it asks "what aspect/type/kind", answer with the specific aspect. If it asks "what did they discover they both enjoy", answer with the specific thing, not the setting.

Step 5: TEMPORAL GROUNDING
These conversations took place around {reference_date}. All events occurred in 2022-2024.
- Calculate time relative to this date, NOT today. Never output 2025 or 2026.
- Use dates explicitly stated in memory text. Do not invent or estimate dates.
- When a question asks what someone "shared" or "mentioned" on a date, that date is when they TALKED about it — look for events shortly BEFORE that date.
- For "how long" questions, find the start and end dates explicitly, then compute the duration. Do not guess.
- TEMPORAL DISAMBIGUATION: When you find MULTIPLE instances of similar events at different dates, enumerate them all with their dates before picking. If the question uses past tense + "the" → select the instance closest to (and before) the reference date. If future tense ("plans to", "going to") → select the earliest planned date. NEVER default to the first-mentioned or highest-scored instance — the DATE determines the answer.

Step 6: INCLUSION CHECK (for lists and counts)
If you found items during reasoning that you're tempted to exclude from your answer — STOP. Include them unless you have STRONG evidence they are wrong. The most common mistake is finding relevant items but then dropping them due to overly strict filtering. More items is better than fewer when there is supporting evidence.
- For counting: after enumerating, re-verify each item. Check for duplicates (same event described differently) and ensure you haven't missed items from memories late in the list.
- The question assumes something happened. Find WHAT happened, don't say nothing happened.

Step 7: COMMIT AND ANSWER
Give a direct, specific answer. NEVER say "not specified", "not mentioned", "no record", or "the memories don't say" — if ANY memory contains relevant information, give the best answer from available evidence. No hedging, no caveats. If the question asks for a list, include ALL items found. NEVER return an empty answer when relevant memories exist.
- NEVER generate specific names, titles, places, or dates that do not appear in any memory above. If no memory contains the specific detail the question asks for, answer with what the memories DO contain rather than guessing.
- For open-domain/opinion questions ("Would X do Y?", "Is X considered Z?"):
  * Follow the DIRECT causal reasoning in the memories. Do NOT construct elaborate counter-arguments.
  * "Would X still do Y without Z?" — If memories show X does Y BECAUSE of Z, then without Z, answer "likely no."
  * "Would X do Y again soon?" — If the most recent attempt involved a bad experience (accident, scare, trauma), answer "likely no." A recent negative experience outweighs historical positive patterns.
  * For trait questions ("Is X considered Z?"): weigh ALL evidence including symbolic/indirect references. If there is SOME but not strong evidence, answer with a qualified degree ("somewhat") rather than flat "no."

Instructions

Misc

1. Make reasonable deductions based on your memories. Memory shows store with a lot of working people -> store employs a lot of people
2. If a memory describes something recognizable (e.g., "romantic drama about memory and relationships"), you may name it (e.g., "Eternal Sunshine of the Spotless Mind").
3. Use domain knowledge to connect facts: a game exclusive to one platform implies ownership of that platform. An unnamed company deal can be linked to a previously expressed brand preference.

{memories}

Question: {question}

Work through Steps 1-7, then give your final answer after "ANSWER:".

The same correctness judge is used for categories 1–4; the category argument does not change the template. The system prompt is empty. The judge receives the question, gold answer, and generated answer through the placeholders below. For open-domain questions, the preprocessing function retains the portion of the gold answer before the first semicolon.

LoCoMo — Answer correctness judge

Your task is to label an answer to a question as 'CORRECT' or 'WRONG'. You will be given the following data:
(1) a question (posed by one user to another user),
(2) a 'gold' (ground truth) answer,
(3) a generated answer
which you will score as CORRECT/WRONG.

The point of the question is to ask about something one user should know about the other user based on their prior conversations.

LoCoMo — Answer correctness judge (continued)

The gold answer will usually be a concise and short answer that includes the referenced topic, for example:
Question: Do you remember what I got the last time I went to Hawaii?
Gold answer: A shell necklace
The generated answer might be much longer, but you should be generous with your grading - as long as it touches on the same topic as the gold answer, it should be counted as CORRECT.

For time related questions, the gold answer will be a specific date, month, year, etc. The generated answer might be much longer or use relative time references (like "last Tuesday" or "next month"), but you should be generous with your grading - as long as it refers to the same date or time period as the gold answer, it should be counted as CORRECT. Even if the format differs (e.g., "May 7th" vs "7 May"), consider it CORRECT if it's the same date.

Now it's time for the real question:
Question: {question}
Gold answer: {gold_answer}
Generated answer: {generated_answer}

First, provide a short (one sentence) explanation of your reasoning, then finish with CORRECT or WRONG.
Do NOT include both CORRECT and WRONG in your response, or it will break the evaluation script.

Just return the label CORRECT or WRONG in a json format with the key as "label".

Optional source-evidence branch. The default `get_judge_prompt` call supplies no source evidence. The separate `get_judge_prompt_with_evidence` entry point inserts the following component immediately before "Now it's time for the real question:" in the judge template.

LoCoMo — Optional judge evidence component

Evidence (actual conversation messages containing the answer)
{evidence_context}

A.4.2 LongMemEval

For LongMemEval, we use the same answer-generation and answer-correctness judging prompts as the Mem0 implementation (Chhikara et al., 2025).

A.4.3 BEAM

For BEAM, we use prompts consistent with the official paper and code implementation (Tavakoli et al., 2026).

B THEORETICAL ANALYSIS

This appendix proves Proposition 1 and derives an optimality certificate for the same selection–refit procedure. Throughout, the query q and candidate pool $\mathcal{C}$ are fixed, embeddings have unit norm, and each refit solves Eq. 6 exactly with a fixed $\lambda \geq 0$. Selected memories remain in the returned context even when their fitted weights become zero.

B.1 SETUP AND RESTRICTED OPTIMALITY

Index coefficient vectors by the P candidate memories and define the full simplex and its restriction to a nonempty set S:

$$\Delta_{\mathcal{C}} = \left\{ \boldsymbol{\alpha} \in \mathbb{R}^P : \boldsymbol{\alpha} \succeq 0,\ \sum_{m \in \mathcal{C}} \alpha_m = 1 \right\}, \qquad \Delta_{\mathcal{C}}(S) = \{ \boldsymbol{\alpha} \in \Delta_{\mathcal{C}} : \alpha_m = 0 \text{ for } m \notin S \}. \tag{12}$$

Write the reconstruction objective as

$$F_\lambda(\boldsymbol{\alpha}; q) = \frac{1}{2} \left\| f(q) - \sum_{m \in \mathcal{C}} \alpha_m f(m) \right\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\alpha}\|_2^2, \qquad J_\lambda(S; q) = \min_{\boldsymbol{\alpha} \in \Delta_{\mathcal{C}}(S)} F_\lambda(\boldsymbol{\alpha}; q). \tag{13}$$

Let $\boldsymbol{\alpha}^S$ be a minimizing coefficient vector, with entries $\alpha_m(q, S)$ on S and zeros outside S. For $S = S_t$, it is the zero-padded version of the refit vector $\boldsymbol{\alpha}_t$ in Eq. 6. A minimizer exists by compactness and is unique for $\lambda > 0$; for $\lambda = 0$, any minimizer may be used. To shorten the proof, write $\hat{\mathbf{q}}_S = \hat{\mathbf{q}}(q, S)$, $\mathbf{r}_S = \mathbf{r}(q, S)$, and

$$U_{\mathrm{MP}}(m \mid q, S) = f(m)^\top \mathbf{r}_S, \qquad \nu_S = \hat{\mathbf{q}}_S^\top \mathbf{r}_S - \lambda \|\boldsymbol{\alpha}^S\|_2^2. \tag{14}$$

Since $\hat{\mathbf{q}}_S$ is a convex combination of unit vectors, $\|\hat{\mathbf{q}}_S\|_2 \leq 1$ and $\|\mathbf{r}_S\|_2 \leq 2$.

Lemma B.1 (Restricted optimality)**.** *For every nonempty S, there are multipliers $\mu_m \geq 0$ such that*

$$U_{\mathrm{MP}}(m \mid q, S) = \nu_S + \lambda \alpha_m^S - \mu_m, \qquad \mu_m \alpha_m^S = 0 \quad (m \in S). \tag{15}$$

In particular,

$$\max_{m \in S} U_{\mathrm{MP}}(m \mid q, S) = \nu_S + \lambda \|\boldsymbol{\alpha}^S\|_\infty. \tag{16}$$

Proof. On the coordinates in S, the Lagrangian is

$$\mathcal{L}(\boldsymbol{\alpha}, \nu, \boldsymbol{\mu}) = F_\lambda(\boldsymbol{\alpha}; q) + \nu \left(\sum_{m \in S} \alpha_m - 1 \right) - \sum_{m \in S} \mu_m \alpha_m.$$

The restricted simplex has a strictly positive feasible vector. Convexity and the KKT conditions give $-U_{\mathrm{MP}}(m \mid q, S) + \lambda \alpha_m^S + \nu - \mu_m = 0$, with $\mu_m \geq 0$ and $\mu_m \alpha_m^S = 0$. Multiplying by α_m^S and summing yields $\nu = \nu_S$, proving Eq. 15. Every selected score is at most $\nu_S + \lambda \|\boldsymbol{\alpha}^S\|_\infty$. A coordinate attaining the largest weight is positive, so its multiplier vanishes and its score attains this bound. □

B.2 FACT-CLUSTER MODEL

The following model links memories supplying the same required fact to their embedding geometry. Its labels and parameters are used for analysis; selection uses only the embeddings and fitted residual.

Assumption B.1 (Fact-cluster separation and coverage margin)**.** The query requires $h \geq 2$ atomic facts, and the candidate memories admit a disjoint partition

$$\mathcal{C} = \mathcal{C}_1 \,\dot\cup\, \cdots \,\dot\cup\, \mathcal{C}_h \,\dot\cup\, \mathcal{D}, \qquad \mathcal{R} = \bigcup_{\ell=1}^{h} \mathcal{C}_\ell.$$

Each $\mathcal{C}_\ell$ is nonempty and contains alternative memories supplying the same required fact; one representative suffices to make that fact available in the context. The remaining candidates form the distractor set $\mathcal{D}$. For $0 \leq \rho < 1$, $\varepsilon, \eta \geq 0$, $a_{\min} \leq a_{\max}$, and $\gamma > 0$, assume:

1. **Within-fact concentration:** $\|f(m) - f(s)\|_2 \le \varepsilon$ for $m, s \in \mathcal{C}_\ell$.
2. **Between-fact similarity:** $|f(m)^\top f(s) - \rho| \le \eta$ for $m \in \mathcal{C}_\ell$, $s \in \mathcal{C}_k$, $\ell \ne k$.
3. **Balanced query relevance:** $a_{\min} \le f(q)^\top f(m) \le a_{\max}$ for $m \in \mathcal{R}$.
4. **Distractor separation:** $f(q)^\top f(d) \le a_{\min} - \gamma$ and $f(d)^\top f(m) \ge \rho - \eta$ for $d \in \mathcal{D}$, $m \in \mathcal{R}$.

Write $\Delta_a = a_{\max} - a_{\min}$. The coverage margin satisfies

$$M := \frac{1-\rho}{h-1} - \Delta_a - 2\eta - 2\varepsilon > 0, \qquad \gamma > 2\eta \text{ when distractors are present.} \tag{17}$$

When $\mathcal{D} = \varnothing$, conditions involving γ are omitted.

B.3 Proof of Proposition 1

Proof. **Geometric redundancy.** The finite convex hull H_S is compact. If $\operatorname{dist}(f(m), H_S) \le \delta$, choose $\mathbf{v} = \sum_{s\in S} \beta_s f(s) \in H_S$ with $\|f(m) - \mathbf{v}\|_2 \le \delta$, where $\beta_s \ge 0$ and $\sum_s \beta_s = 1$. Then

$$\begin{aligned} U_{\mathrm{MP}}(m \mid q, S) &= \sum_{s\in S} \beta_s U_{\mathrm{MP}}(s \mid q, S) + (f(m) - \mathbf{v})^\top \mathbf{r}_S \\ &\le \max_{s\in S} U_{\mathrm{MP}}(s \mid q, S) + \delta \|\mathbf{r}_S\|_2 \\ &= \nu_S + \lambda \|\boldsymbol{\alpha}^S\|_\infty + \delta \|\mathbf{r}_S\|_2. \end{aligned} \tag{18}$$

The inequality uses Cauchy–Schwarz and the equality uses Lemma B.1. This proves Eq. 8. If an unselected candidate exceeds this bound, no memory within distance δ of H_S can maximize the selection score.

Coverage induction. At the first step, each required-fact memory has query similarity at least $a_{\min}$, whereas every distractor has similarity at most $a_{\min} - \gamma < a_{\min}$. Thus the first selection belongs to a required-fact cluster. If there are no distractors, this holds directly.

Suppose $S = S_t$ contains one representative from each of $t < h$ distinct fact clusters and no distractor. Let G_{SS} be the selected memory Gram matrix. Assumption B.1 and unit norms give

$$G_{SS} = (1-\rho) I_t + \rho \mathbf{1}\mathbf{1}^\top + E, \qquad E_{ss} = 0, \quad |E_{ms}| \le \eta. \tag{19}$$

Write $\boldsymbol{\alpha} = \boldsymbol{\alpha}^S|_S$. Simplex feasibility gives $|(E\boldsymbol{\alpha})_s| \le \eta$ and $\alpha_{\max} \ge 1/t$. Choose a selected memory j attaining $\alpha_{\max}$. By Lemma B.1, it also attains the largest selected residual score, so

$$\begin{aligned} \max_{s\in S} U_{\mathrm{MP}}(s \mid q, S) &= f(q)^\top f(j) - \rho - (1-\rho)\alpha_{\max} - (E\boldsymbol{\alpha})_j \\ &\le a_{\max} - \rho + \eta - \frac{1-\rho}{t}. \end{aligned} \tag{20}$$

For an unselected memory m from an already represented cluster, its selected representative lies within distance ε. Equation 18 and $\|\mathbf{r}_S\|_2 \le 2$ therefore imply

$$U_{\mathrm{MP}}(m \mid q, S) \le a_{\max} - \rho + \eta - \frac{1-\rho}{t} + 2\varepsilon. \tag{21}$$

For a memory u from an uncovered cluster, every selected memory belongs to a different cluster. Hence $f(u)^\top \hat{\mathbf{q}}_S \le \rho + \eta$ and

$$\begin{aligned} U_{\mathrm{MP}}(u \mid q, S) &\ge a_{\min} - \rho - \eta, \\ U_{\mathrm{MP}}(u \mid q, S) - U_{\mathrm{MP}}(m \mid q, S) &\ge \frac{1-\rho}{t} - \Delta_a - 2\eta - 2\varepsilon \ge M > 0. \end{aligned} \tag{22}$$

For a distractor d, the separation assumption gives $f(d)^\top \hat{\mathbf{q}}_S \ge \rho - \eta$, yielding

$$\begin{aligned} U_{\mathrm{MP}}(d \mid q, S) &\le a_{\min} - \gamma - \rho + \eta, \\ U_{\mathrm{MP}}(u \mid q, S) - U_{\mathrm{MP}}(d \mid q, S) &\ge \gamma - 2\eta > 0. \end{aligned} \tag{23}$$

An uncovered cluster exists because $t < h$. Every memory from such a cluster strictly outranks both redundant candidates and distractors, so the next selection represents a new required fact. Induction gives one representative from each of the h clusters in the first h selections. Since selected memories are retained, coverage persists at every larger returned prefix. □

The coverage conclusion concerns the availability of required facts in the selected memory texts. It holds for every fixed $\lambda \geq 0$ under Eq. 17, while allowing the fitted weights and selection order to depend on λ.

B.4 A COMPUTABLE OPTIMALITY CERTIFICATE

The following certificate specializes the Frank–Wolfe gap (Jaggi, 2013) to the ridge-regularized simplex refit. It requires only the setup in Appendix B.1. Define the optimum over the full candidate simplex by

$$F_\lambda^\star(q) = \min_{\boldsymbol{\alpha} \in \Delta_\mathcal{C}} F_\lambda(\boldsymbol{\alpha}; q). \tag{24}$$

Theorem B.2 (Computable optimality certificate)**.** *For a nonempty selected set S with exact refitting, define*

$$\begin{aligned} g_S &= \left[\max_{m \in \mathcal{C} \setminus S} U_{\mathrm{MP}}(m \mid q, S) - \nu_S \right]_+ \\ &= \left[\max_{m \in \mathcal{C} \setminus S} f(m)^\top \mathbf{r}_S - \hat{\mathbf{q}}_S^\top \mathbf{r}_S + \lambda \|\boldsymbol{\alpha}^S\|_2^2 \right]_+, \end{aligned} \tag{25}$$

where $[a]_+ = \max\{a, 0\}$ and $\max \varnothing = -\infty$. Then g_S is the Frank–Wolfe gap of $F_\lambda(\cdot; q)$ on $\Delta_\mathcal{C}$ at $\boldsymbol{\alpha}^S$, and

$$0 \leq J_\lambda(S; q) - F_\lambda^\star(q) \leq g_S. \tag{26}$$

Moreover, $g_S = 0$ if and only if $\boldsymbol{\alpha}^S$ minimizes the objective over the full simplex.

Proof. Let $\mathbf{p} = \nabla_{\boldsymbol{\alpha}} F_\lambda(\boldsymbol{\alpha}^S; q)$. Its coordinates satisfy $p_m = -U_{\mathrm{MP}}(m \mid q, S) + \lambda \alpha_m^S$, and $(\boldsymbol{\alpha}^S)^\top \mathbf{p} = -\nu_S$. By Lemma B.1, $p_m \geq -\nu_S$ on S, with equality at every positive-weight coordinate. At least one such coordinate exists, so $\min_{m \in S} p_m = -\nu_S$. Outside S, the weights vanish and $p_m = -U_{\mathrm{MP}}(m \mid q, S)$. Consequently,

$$\begin{aligned} g_{\mathrm{FW}}(\boldsymbol{\alpha}^S) &:= \max_{\boldsymbol{\beta} \in \Delta_\mathcal{C}} \langle \mathbf{p}, \boldsymbol{\alpha}^S - \boldsymbol{\beta} \rangle \\ &= (\boldsymbol{\alpha}^S)^\top \mathbf{p} - \min_{m \in \mathcal{C}} p_m \\ &= -\nu_S - \min \left\{ -\nu_S, - \max_{m \in \mathcal{C} \setminus S} U_{\mathrm{MP}}(m \mid q, S) \right\} = g_S. \end{aligned} \tag{27}$$

Let $\boldsymbol{\alpha}^\star$ be a full-simplex minimizer. Convexity gives

$$F_\lambda(\boldsymbol{\alpha}^\star; q) \geq F_\lambda(\boldsymbol{\alpha}^S; q) + \langle \mathbf{p}, \boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^S \rangle.$$

Rearranging and maximizing the resulting inner product over the simplex proves Eq. 26. Thus $g_S = 0$ implies optimality. Conversely, first-order optimality at a full-simplex minimizer gives $\langle \mathbf{p}, \boldsymbol{\beta} - \boldsymbol{\alpha}^S \rangle \geq 0$ for every feasible $\boldsymbol{\beta}$; choosing $\boldsymbol{\beta} = \boldsymbol{\alpha}^S$ shows that the maximum defining the gap is zero. □

The certificate bounds the remaining error in the regularized reconstruction objective over the current candidate pool. It is computed from the residual scores and fitted weights already used by MEMPURSUIT.

C Additional Experimental Results

C.1 Full cross-framework results

Table 4 reports the complete results at $K = 10$ and $K = 20$ for the cross-framework comparisons in Table 3.

Table 4: Full cross-framework comparisons on LoCoMo at $K \in \{10, 20\}$. Each system is evaluated without (w/o) and with (w/) MemPursuit at the same selection budget K (seed count for MemoryBank). Entries report accuracy (%); parentheses in the w/ rows give the change in percentage points from the paired w/o result. For each system and budget, the higher accuracy in each metric is bolded. Yellow highlights the w/ results. All configurations evaluate 1,540 questions using GPT-4o-mini as answerer and judge.

K	System	MemPursuit	Multi-hop	Temporal	Open-domain	Single-hop	Overall
10	LightMem	w/o	62.77	66.98	56.25	77.17	71.10
		w/	**69.15** (+6.38)	**68.22** (+1.25)	**57.29** (+1.04)	**79.19** (+2.02)	**73.70** (+2.60)
	MemoryBank	w/o	53.55	**43.30**	51.04	65.87	57.99
		w/	**56.74** (+3.19)	39.25 (−4.05)	**56.25** (+5.21)	**70.27** (+4.40)	**60.45** (+2.47)
	SimpleMem	w/o	62.41	48.60	56.25	74.91	65.97
		w/	**70.21** (+7.80)	**49.22** (+0.62)	**60.42** (+4.17)	**82.28** (+7.37)	**71.82** (+5.84)
	Mem0	w/o	45.04	42.37	43.75	57.43	51.17
		w/	**55.32** (+10.28)	**49.84** (+7.48)	**48.96** (+5.21)	**67.54** (+10.11)	**60.45** (+9.29)
20	LightMem	w/o	73.40	**70.09**	**63.54**	83.12	77.40
		w/	**74.11** (+0.71)	**70.09** (+0.00)	61.46 (−2.08)	**83.71** (+0.59)	**77.73** (+0.32)
	MemoryBank	w/o	59.22	**48.91**	**54.17**	72.89	64.22
		w/	**64.18** (+4.96)	47.66 (−1.25)	**54.17** (+0.00)	**78.72** (+5.83)	**68.05** (+3.83)
	SimpleMem	w/o	70.21	52.96	**67.71**	79.90	71.75
		w/	**77.66** (+7.45)	**54.83** (+1.87)	64.58 (−3.12)	**85.97** (+6.06)	**76.62** (+4.87)
	Mem0	w/o	56.38	47.04	**51.04**	63.85	58.18
		w/	**58.51** (+2.13)	**51.71** (+4.67)	50.00 (−1.04)	**69.32** (+5.47)	**62.47** (+4.29)

C.2 Reconstruction Residuals and Gold Evidence

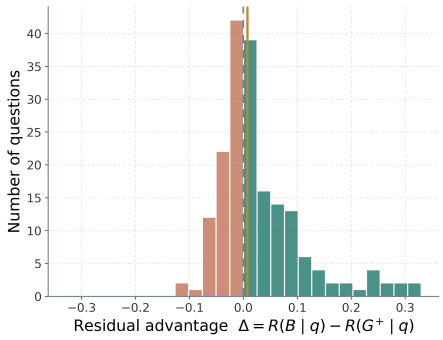


Figure 7: Reconstruction residual advantage of gold evidence over hybrid retrieval at matched set size. Positive Δ_q (green) indicates a lower residual norm for the gold evidence set; negative Δ_q (orange) indicates a lower residual for the baseline. The dashed line marks equal residuals. Both sets are evaluated with the same simplex refit.

Reconstruction provides a useful target for evidence composition. To connect the evidence-coverage gains in Section 4.3 to the selection objective, we compare the reconstruction residuals of gold evidence and hybrid retrieval at equal set size. For each query q, let G_q^+ be its gold evidence memory set and B_q the hybrid Top-K_q^{gold} set, where $K_q^{\text{gold}} = |G_q^+|$. We keep the gold set fixed and match the baseline budget to it. We evaluate both sets using the refitted residual norm $R(S \mid q) = \|\mathbf{r}(q, S)\|_2$, testing whether gold evidence yields a closer fit in the same representation space that supplies the residual-guided utility in Eq. 4. We report the paired residual advantage

$$\Delta_q = R(B_q \mid q) - R(G_q^+ \mid q), \tag{28}$$

so positive values indicate that gold evidence reconstructs the query more closely than the equally sized hybrid selection.

Figure 7 shows that gold evidence achieves lower residuals than hybrid Top-K_q^{gold} on many queries, with pronounced advantages on the positive side of the distribution. At equal set size, this advantage links the content of the evidence to how closely its embeddings

reconstruct the query. It provides an empirical basis for MEMPURSUIT's central mechanism: refitting accounts for what the selected memories already represent, and the residual directs subsequent recall toward query content that remains uncovered. Together with the coverage gains, this analysis connects the geometry used for selection to the complementary evidence recovered in the answer context.

C.3 RESIDUAL PREFERENCE FOR MISSING EVIDENCE

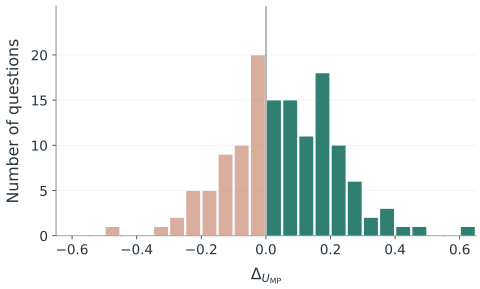


Figure 8: Residual preference for missing evidence on LongMemEval with $|S| = 1$. The histogram contains one mean utility difference per question (136 questions, 381 singleton comparisons). Positive values (green) favor remaining evidence over the semantic comparator, while negative values (orange) favor the comparator. The gray line marks zero.

We test whether the residual score favors missing evidence once an answer-bearing memory is already selected. We reuse saved $P = 200$ candidate pools from LongMemEval for multi-session, temporal-reasoning, and knowledge-update questions with at least two annotated answer sessions, all present in the pool. Because session annotations do not identify which individual memories contain answer facts, a local Qwen3.5-4B judge receives the question, reference answer, and candidate memories from the annotated sessions. A memory is accepted as evidence only if the judge identifies a required answer fact and supplies a supporting quotation that occurs verbatim in the memory. We retain 136 questions with evidence in at least two answer sessions and missing evidence remaining after duplicate-fact filtering.

For each question, we rotate the accepted evidence memories through the singleton selected set $S^{(j)} = \{s^{(j)}\}$, yielding 381 comparisons in total. We remove evidence repeating the selected fact and keep one randomly sampled representative per remaining duplicate-fact group, using supporting-quotation overlap and embedding similarity to identify duplicates. The semantic comparator $m^{(j)}_{\text{sem}}$ is the highest-query-cosine candidate after excluding $s^{(j)}$ and memories supplying an unselected answer fact. It may therefore repeat the selected fact or be a candidate not accepted as evidence. We compute U_{MP} using Eq. 4. Remaining-evidence scores are averaged within each answer session and then equally across sessions to obtain $U^{(j)}_E$. Each question contributes one mean difference:

$$\overline{\Delta}_{U_{\text{MP}}}(q) = \frac{1}{D_q} \sum_{j=1}^{D_q} \left[U^{(j)}_E - U_{\text{MP}}(m^{(j)}_{\text{sem}} \mid q, S^{(j)}) \right], \tag{29}$$

where D_q is the number of singleton selections for question q.

Figure 8 shows more questions with positive than negative utility differences. Once one evidence memory is selected, residual scoring tends to favor complementary answer facts over highly query-similar alternatives. This controlled comparison supports the role of the residual in directing selection toward evidence still missing from the context.

Case study: Recovering missing evidence. Table 5 traces this behavior in a LongMemEval multi-session question. At the same budget of ten memories, hybrid retrieval includes repeated descriptions of the grandmother's 75th birthday but leaves the user's age just outside the cutoff, at rank 11. MEMPURSUIT retains the grandmother's age and selects the user's age at step 6, completing the evidence needed to compute $75 - 32 = 43$. Both answerers then return the correct age difference. This example complements the reconstruction analysis in Appendix C.2 and the residual-utility comparison above by tracing missing evidence into the selected context and the resulting answer.

Table 5: Recovery of a missing age fact on LongMemEval ($K = 10$). Memory excerpts show the two required facts and a repeated birthday description. Hybrid positions are retrieval ranks, while MEMPURSUIT positions are greedy selection steps. Answers are reported as age differences in years.
Query: How many years older is my grandma than me?
Gold answer: 43.

Memory excerpt / outcome	**Hybrid baseline**	**MEMPURSUIT**
"User's grandma is energetic and sharp at **75 years old**, and user has been thinking about her a lot lately…"	Rank 1 (selected)	Step 1 (selected)
"User's grandma's **75th birthday celebration** was a significant event for their family, inspiring them to plan a similar gathering."	Rank 6 (selected)	Outside Top-10
"User is **32 years old** and is reflecting on whether this age is considered young or old in the grand scheme of things."	**Rank 11** (outside Top-10)	**Step 6** (selected)
Required age facts in the context	1/2	**2/2**
Qwen3-30B-A3B answer	1 (incorrect)	**43** (correct)
GPT-4o-mini answer	52 (incorrect)	**43** (correct)

Recovering the earlier event in a temporal comparison. In the LoCoMo example in Table 6, hybrid retrieval includes records about Buddy and the two dogs but misses Toby's adoption in July. MEMPURSUIT selects this earlier event at step 3, promoting it from hybrid rank 26. Together with the October record of Buddy's recent adoption, it supplies both endpoints of the approximately three-month interval, allowing Qwen3-30B-A3B to answer correctly at $K = 10$.

Table 6: Recovering a missing adoption event on LoCoMo (`conv5_q34`, temporal, $K = 10$). Hybrid ranks and pursuit steps refer to the same memories. The date outside quotation marks is the memory timestamp. Answers are condensed.
Query: How many months passed between Andrew adopting Toby and Buddy?
Gold answer: Three months.

Memory excerpt / outcome	**Hybrid baseline**	**MEMPURSUIT**
"Andrew recently adopted a new dog named **Buddy** from a shelter…" *Recorded: 19 October 2023.*	Rank 7 (selected)	Step 1 (selected)
"Andrew adopted a puppy named **Toby**, bringing him home to enjoy city living around **July 2023**."	**Rank 26** (outside Top-10)	**Step 3** (selected)
Qwen3-30B-A3B answer	Cannot determine (incorrect)	**3 months** (correct)

Recovering a specific diagnosis beyond general health concerns. Table 7 illustrates a different omission. The hybrid context contains Sam's weight-related health warning but excludes the memory naming his diagnosis. MEMPURSUIT retains the warning and promotes the gastritis record from hybrid rank 41 to step 6. The answer consequently changes from a general health risk to the specific condition named in the benchmark answer. This case concerns recovery of the annotated diagnosis, rather than establishing a medical causal relationship between weight and gastritis.

Table 7: Recovering the annotated diagnosis on LoCoMo (`conv8_q64`, multi-hop, $K = 10$). Memory excerpts distinguish the general health warning from the specific diagnosis. Correctness follows the benchmark annotation.
Query: Which ailment does Sam have to face due to his weight?
Gold answer: Gastritis.

Memory excerpt / outcome	**Hybrid baseline**	**MEMPURSUIT**
"Sam went for a health check-up on October 2, 2023, where his doctor informed him that his weight poses a **serious health risk**…"	Rank 4 (selected)	Step 3 (selected)
"Sam had a health scare last weekend, ending up in the ER with a severe stomachache, which was diagnosed as **gastritis**."	**Rank 41** (outside Top-10)	**Step 6** (selected)
Qwen3-30B-A3B answer	Serious health risk (incorrect)	**Gastritis** (correct)

Recovering the second date within the same source batch. In the BEAM example in Table 8, hybrid retrieval repeatedly selects meetup-related memories but omits the survey date at rank 12. MEMPURSUIT selects the March 28 meetup and the March 10 survey at steps 1 and 2, respectively. Both answerers then return the correct interval of 18 days. The two memories belong to the same source batch, illustrating why source coverage alone can miss an omission of a required fact.

Table 8: Recovering a missing date on BEAM-500K (conversation 32, temporal reasoning question 1, $K = 10$). Both memory excerpts originate from `batch_001`. Event dates appear in the memory text. Answers are condensed.
Query: How many days do I have between finalizing my survey plans at Café Soleil and attending the Raymondburgh Startup Meetup to prepare effectively?
Gold answer: 18 days.

Memory excerpt / outcome	**Hybrid baseline**	**MEMPURSUIT**
"User plans to attend the Raymondburgh Startup Meetup on **March 28, 2024**, to meet potential mentors…"	Rank 1 (selected)	Step 1 (selected)
"User is concerned that their survey at Café Soleil on **March 10, 2024**, may not yield the necessary results…"	**Rank 12** (outside Top-10)	**Step 2** (selected)
Qwen3-30B-A3B answer	Cannot determine (incorrect)	**18 days** (correct)
GPT-4o-mini answer	30 days (incorrect)	**18 days** (correct)

C.4 QUESTION-TYPE RESULTS AT TOP-20

Table 9: LoCoMo question-type accuracy (%) at Top-20 under the controlled LoCoMo evaluation setup, comparing hybrid ordering, MEMPURSUIT, and pointwise neural rerankers.

Read policy	Multi-hop	Temporal	Open-domain	Single-hop	Overall
Original hybrid order	75.5	73.8	64.6	84.9	79.6
MEMPURSUIT	80.9	77.9	63.5	90.7	85.1
Qwen3.5-0.8B	78.0	77.9	63.5	87.8	82.4
Qwen3.5-2B	81.6	79.4	66.7	91.3	85.5
Qwen3.5-4B	86.9	80.7	65.6	91.1	86.6
BGE-base	75.2	67.6	64.6	87.2	79.5
BGE-large	81.2	78.8	65.6	91.9	85.6
BGE-v2-m3	85.1	80.4	64.6	92.9	87.1

C.5 COMPARISON WITH MAXIMAL MARGINAL RELEVANCE

We compare MEMPURSUIT with maximal marginal relevance (MMR) (Carbonell & Goldstein, 1998), which balances query relevance against the maximum similarity to an already selected item. While MMR explicitly penalizes pairwise redundancy, MEMPURSUIT optimizes a joint query-reconstruction objective over the selected set. Residual scoring and joint refitting work together to select memories that improve this reconstruction. This comparison tests whether reconstruction-driven composition improves on explicit redundancy penalization.

We use the same LoCoMo memory store and per-question candidate pool of $P = 200$ memories, with GPT-4o-mini as both answerer and judge, and vary only the memory selection strategy. MMR uses a fixed relevance–diversity coefficient $\lambda_{\mathrm{MMR}} = 0.5$. Each budget $K \in \{10, 20, 50, 200\}$ is evaluated on 1,540 questions.

Table 10: LoCoMo accuracy (%) for MEMPURSUIT and MMR with $\lambda_{\mathrm{MMR}} = 0.5$, using the same $P = 200$ candidate pool and downstream evaluation. Each row contains 1,540 questions; bold denotes the higher accuracy.

Memory budget K	MEMPURSUIT	MMR
10	**81.04**	77.08
20	**85.06**	83.31
50	**87.08**	86.10

MEMPURSUIT achieves higher accuracy at all three tested budgets (Table 10). The largest advantage occurs at Top-10, where accuracy increases from 77.08% to 81.04%, a gain of 3.96 percentage points; at Top-20, the gain is 1.75 points. These results show the value of residual-guided composition over MMR at the tested coefficient, particularly when only a small fraction of the candidate memories can enter the answer context.

C.6 ROBUSTNESS TO EMBEDDING CHOICE

We compare five encoders on LoCoMo at Top-10, Top-20, and Top-50, keeping each question's candidate memories fixed and changing only the embeddings used for composition. The encoders are text-embedding-3-small, E5-large-v2, BGE-M3, Qwen3-Embedding-0.6B, and Qwen3-Embedding-8B. Each configuration evaluates 1,540 questions under the same downstream evaluation setup.

Table 11 reports the results. All tested encoders outperform the fixed hybrid baseline at these budgets. E5 and BGE remain close to the default encoder, while Qwen3-Embedding-8B further improves accuracy at Top-10 and Top-20. These results demonstrate the applicability of MEMPURSUIT across representation families and its ability to benefit from stronger embeddings under tight budgets.

Table 11: Robustness to embedding choice on LoCoMo with identical per-question candidate memories. Only the dense encoder used by MEMPURSUIT changes. Entries report accuracy (%) over 1,540 questions; parentheses give changes in percentage points relative to text-embedding-3-small. Bold denotes the highest accuracy at each cutoff.

Embedding model	Top-10	Top-20	Top-50
text-embedding-3-small	81.04	85.06	**87.08**
E5-large-v2	80.32 (−0.71)	83.57 (−1.49)	86.49 (−0.58)
BGE-M3	80.00 (−1.04)	84.68 (−0.39)	86.10 (−0.97)
Qwen3-Embedding-0.6B	79.94 (−1.10)	82.40 (−2.66)	85.65 (−1.43)
Qwen3-Embedding-8B	**83.18** (+2.14)	**85.91** (+0.84)	86.49 (−0.58)

C.7 READ-TIME EFFICIENCY

Hardware and execution. The experimental server has two Intel Xeon Gold 5117 CPUs at a base frequency of 2.00 GHz, with 28 physical cores and 56 hardware threads in total, and 256 GiB of RAM, running Ubuntu 20.04.5 LTS. MEMPURSUIT runs on the CPU with a Numba backend and a single worker, with BLAS and OpenMP thread counts set to one. Each neural reranking run uses a single NVIDIA GeForce RTX 4090 GPU with 24 GB of device memory. Qwen3.5 rerankers use BF16 precision.

Neural rerankers and latency measurement. We retain the memory store, candidate pool, answerer, and judge from Section 4.3, fixing $P = 200$ and $K = 20$. BGE-base, BGE-large, and BGE-v2-m3 are cross-encoder rerankers; Qwen3.5-0.8B, -2B, and -4B are pointwise LLM scorers. Figure 5 uses batch size 128, a large candidate batch size supported by all evaluated models. Each batch contains candidate memories for a single question. We measure final reranking latency, excluding memory construction, candidate retrieval, and answer generation. The figure reports medians. Speedups based on both mean and median latency, together with tail-latency statistics, are reported below. Full-batch BGE inference scores all 200 candidates for a question in a single batch.

Table 12: Final reranking latency at Top-20 with $P = 200$ candidates. Neural scorers use batch size 128, a large candidate batch size supported by all evaluated BGE and Qwen models. Tail latency is P95 for every method. The two speedup columns divide each baseline's mean or median latency by the corresponding MEMPURSUIT value (17.69 or 8.67 ms).

		Latency (ms)			Speedup	
Read policy	Batch	Mean	Median	P95	Mean latency	Median latency
MEMPURSUIT	–	17.69	8.67	57.76	1.00×	1.00×
BGE-base	128	109.26	109.03	126.12	6.18×	12.58×
BGE-large	128	283.73	283.27	330.01	16.04×	32.67×
BGE-v2-m3	128	275.66	274.33	320.70	15.58×	31.64×
Qwen3.5-0.8B	128	1682.95	1675.98	1751.40	95.14×	193.31×
Qwen3.5-2B	128	2119.66	2113.75	2205.36	119.82×	243.80×
Qwen3.5-4B	128	5151.40	5138.73	5343.48	291.20×	592.70×

MEMPURSUIT takes 27.24 s in total over the reported evaluation run. Table 12 uses the same batch size 128 as Figure 5; each batch contains candidate memories for a single question. Figure 9 reports the batch-size sweeps for both cross-encoder and LLM rerankers, with MEMPURSUIT shown at its median selection latency of 8.67 ms.

Accuracy gain relative to hybrid retrieval. We measure the fraction of a reranker's accuracy gain recovered by MEMPURSUIT relative to the same hybrid-retrieval baseline. Using the displayed accuracies, this fraction is $(85.1 - 79.6)/(87.1 - 79.6) \approx 73\%$ for BGE-v2-m3. At batch size 128, the corresponding ratio of median reranking latencies is $8.67/274.329 \approx 3.2\%$. The gain fraction compares improvements in answer accuracy; the latency fraction compares median final-reranking time per question.

Full-batch comparison. For BGE-v2-m3, full-batch inference scores all $P = 200$ candidates for a question in a single batch. Its median reranking latency is 285.612 ms, compared with 8.67 ms for MEMPURSUIT, yielding the $285.612/8.67 \approx 32.9\times$ speedup in median latency reported in Section 4.5 and the abstract. Table 13 extends the full-batch comparison to three BGE and two Qwen3.5 scorers, reporting mean, median, and P95 latency together with speedups computed from median latency. These results complement the batch-128 comparison in Figure 5 and Table 12.

Table 13: Full-batch reranking latency at Top-20 with $P = 200$ candidates over 1,540 questions. Neural scorers use batch size 200. Latencies are in milliseconds. "Median speedup" divides each baseline median by the MEMPURSUIT median of 8.67 ms.

Read policy	Mean ms	Median ms	P95 ms	Median speedup
MEMPURSUIT	17.69	8.67	57.76	1.00×
BGE-base	111.34	110.48	129.88	12.74×
BGE-large	286.10	286.02	331.60	32.99×
BGE-v2-m3	286.45	285.61	332.61	32.94×
Qwen3.5-0.8B	1733.86	1724.74	1794.96	198.93×
Qwen3.5-2B	2129.20	2124.70	2222.55	245.06×

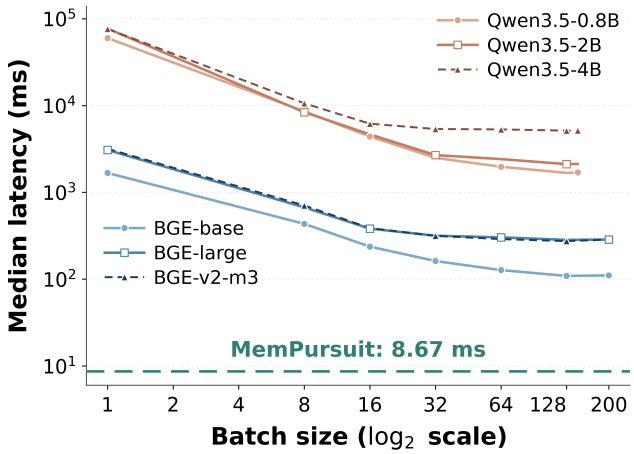


Figure 9: Median reranking latency across batch sizes for BGE cross-encoders (blue) and Qwen3.5 LLM scorers (orange), with MEMPURSUIT as a batch-independent reference at 8.67 ms (green dashed line). The horizontal axis uses a base-2 logarithmic scale and the vertical axis a base-10 logarithmic scale. BGE sweeps extend to batch size 200 and Qwen sweeps to 144.

C.7.1 LATENCY ACROSS CANDIDATE AND CONTEXT BUDGETS

Table 14 reports median reranking latency as we vary the candidate count $P \in \{100, 200, 300, 500\}$ and the selected-memory budget $K \in \{10, 20, 50, 100\}$. Latency increases with both the number of candidates and the number of selected memories, while remaining below 40 ms throughout the tested grid. At $K = 20$, increasing P from 100 to 500 raises the median from 6.30 ms to 22.47 ms. Even the largest tested configuration, $P = 500$ and $K = 100$, requires only 38.58 ms, demonstrating inexpensive selection across a broad range of candidate and context budgets.

Table 14: Median MEMPURSUIT reranking latency (ms per question) across candidate counts P and selected-memory budgets K. Each entry is measured in the P/K sweep.

	Selected-memory budget K			
Candidates P	10	20	50	100
100	4.54	6.30	11.79	22.75
200	7.27	8.40	13.62	23.99
300	9.73	12.16	17.11	26.49
500	18.04	22.47	30.67	38.58