Beyond RAG for Agent Memory: Retrieval by Decoupling and Aggregation

Zhanghao Hu[1,2*] **Qinglin Zhu**[1*] **Runcong Zhao**[1] **Di Liang**[2]
Hanqi Yan[1] **Yulan He**[1] **Lin Gui**[1]

[1]King's College London
[2]Tencent, Yuanbao Team
{zhanghao.hu, qinglin.1.zhu, lin.1.gui}@kcl.ac.uk

Abstract

Standard Retrieval Augmented Generation (RAG) is poorly matched to agent memory. Unlike large heterogeneous corpora, agent memory forms a bounded and coherent interaction stream in which many spans are highly correlated or near duplicates. As a result, flat top-k similarity retrieval often returns redundant context, while summary-centric hierarchies can blur the subtle details that distinguish one candidate from another. We argue that agent memory should follow the principle of *decoupling before aggregation*: the system should first isolate reusable facts, updates, and distinguishing details from similar histories, and only then organise them for efficient retrieval. Based on this principle, we propose xMemory, which constructs a revisable hierarchical memory structure from original messages to segments, memory components, and groups. xMemory segments interaction history into local events, decouples each segment into memory components, aggregates related components into high-level groups using a sparsity–semantic faithfulness objective, and maintains this structure incrementally as memory evolves. At inference time, xMemory retrieves top-down, first selecting a compact backbone of complementary groups and components, and then expanding to segments and raw messages only when additional evidence reduces the reader's uncertainty. Experiments on LoCoMo and PerLTQA across diverse open source and closed source LLMs show consistent gains in answer quality and inference token efficiency, supported by analyses of redundancy, evidence density, and coverage. Code is available at `https://github.com/HU-xiaobai/xMemory`.

1 Introduction

Large language model (LLM) agents increasingly rely on external memory to manage extended interaction histories, enabling capabilities such as multi-session dialogue, personalised assistance, and long-term task execution [1, 2, 3, 4]. In these settings, an agent answers new queries by retrieving useful evidence from memory systems rather than relying only on parametric knowledge [5, 6]. Current systems often approach memory retrieval as a standard RAG problem, relying on fixed top-k similarity [7, 8, 9]. However, agent memory poses a different retrieval challenge from standard RAG corpora. Unlike large heterogeneous document collections, agent memory forms a coherent interaction stream with substantial overlap in events and wording [10, 11]. Consequently, the central retrieval difficulty is no longer merely locating generally relevant text, but rather *distinguishing decisive evidence hidden among highly similar histories*. As a result, similarity top-k retrieval often returns multiple memories that are generally relevant but largely repetitive. This is particularly

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

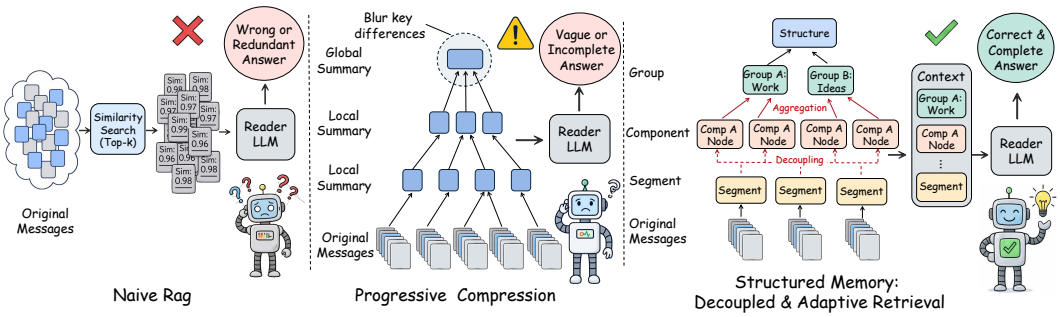

Figure 1: **From similarity top-k retrieval to evidence-oriented retrieval for agent memory.** Agent memory forms a coherent and highly correlated interaction stream, in which many spans are near duplicates. As a result, fixed top-k similarity retrieval often overweights redundant evidence instead of surfacing what is most useful for the query. xMemory first decouples similar histories into memory components that isolate distinctive evidence, and then organises them into a retrieval hierarchy that yields shorter, more sufficient context.

problematic when the answer hinges on a small update, constraint, or factual difference among otherwise similar interactions.

This mismatch also exposes a limitation of existing memory organisation strategies. Recent systems use summaries, note structures, and hierarchical abstractions to improve scalability and navigation [12, 8, 9]. Although these structures reduce the cost of flat retrieval, many still organise memory into a progressively compressed hierarchy, where nodes become fewer and more compressed at higher levels. Such designs can improve efficiency, but they do not directly address the need to identify answer-critical evidence from highly similar histories. Repeated compression often preserves shared background information while blurring the subtle details that distinguish one candidate from another.

Taken together, these considerations suggest a core design principle: *decoupling before aggregation*. As shown in Fig. 1, rather than treating long, highly similar interaction logs as whole retrieval units, the memory system should first decompose them into smaller evidence units that isolate reusable facts, state updates, and distinguishing details. Aggregation then organises these units into a higher-level memory structure that remains compact and coherent. Crucially, this structure should be revisable rather than fixed: as agent memory evolves, new interactions can reveal better relations among previously stored components and trigger corresponding updates to the high-level organisation.

Based on this principle, we propose xMemory, a framework that couples **memory structuring** with **adaptive retrieval**. Starting from raw messages, xMemory builds a hierarchical memory structure by following the principle of *decoupling before aggregation*: it extracts *segments* from raw messages as local events, decouples similar histories into *memory components* that isolate reusable facts and distinctive attributes, and then aggregates related components into *groups* for high-level access. To organise these components into useful groups, xMemory evaluates the component-to-group organisation with a guidance objective that balances sparsity and semantic faithfulness. This objective guides structure updates as memory evolves. When new memory arrives, components are attached to compatible groups when possible, and the structure is revised as needed by splitting overly large or internally heterogeneous groups and merging overly small or isolated ones. Retrieval finally proceeds from coarse units to fine ones: the system first selects complementary groups and components, and expands to segments and original messages only when more detailed evidence is needed.

Our contributions are summarised as follows:

1. We identify a key mismatch between standard RAG and agent memory, and argue that agent memory should follow the principle of *decoupling before aggregation*.
2. We propose xMemory, which constructs a revisable hierarchical memory structure by segmenting local interaction history, decoupling segments into memory components, and aggregating related components into high-level groups.
3. We develop a top-down adaptive retrieval method over this memory structure, and show on LoCoMo and PerLTQA that it improves both answer quality and inference token efficiency.

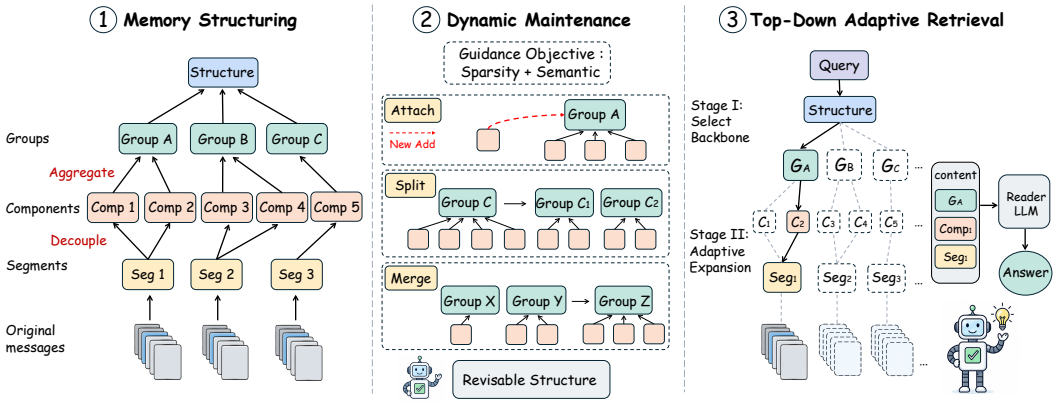

Figure 2: **Overview of xMemory.** xMemory follows the principle of *decoupling before aggregation*. It first organises raw interaction history into local segments, then decomposes each segment into reusable memory components that isolate decisive evidence, and finally aggregates related components into high-level groups for efficient retrieval. At retrieval time, the system first selects a compact backbone of relevant groups and components, and expands to segments and raw messages only when additional textual evidence reduces reader uncertainty.

2 Related Work

RAG-style retrieval for agent memory. Many memory systems follow the standard RAG paradigm, storing past interactions as retrievable units and selecting a top-k set by embedding similarity [6, 13]. This works well for heterogeneous corpora, but agent memory is often a coherent interaction stream with many semantically similar spans [14]. As a result, similarity retrieval can return redundant memories while missing small updates or temporal distinctions that determine the answer. xMemory addresses the mismatch by retrieving decoupled evidence units rather than raw spans.

Hierarchical and graph-based memory organisation. Recent systems organise memory into summaries, notes, temporal layers, semantic memories, or graph communities to improve scalability [15, 8, 9]. While these structures reduce the cost of flat retrieval, many remain summary- or schema-centric: higher-level nodes abstract over lower-level content instead of preserving fine-grained distinctions. xMemory instead follows *decoupling before aggregation*: it first extracts reusable memory components from local segments, and then groups related components for efficient access.

Adaptive retrieval over dynamic memory structures. RAG-style memory systems usually treat past interactions as a static retrieval corpus, accessed by top-k search, neighbour expansion, or multi-stage filtering [7, 13]. Some hierarchical systems support incremental updates by inserting new memories into existing clusters or nodes, but this remains distinct from retroactive reorganisation [16, 17]. This distinction matters for agent memory, where later interactions can change how earlier evidence should be grouped. xMemory therefore maintains a revisable hierarchy that uses split and merge operations to reorganise previously stored components before adaptive retrieval.

3 Method

Given an interaction history $H = \{m_1, \ldots, m_T\}$ and a query q, the goal is to construct a context C for answering q. As shown in Figure 2, xMemory first organises the interaction history into a hierarchical memory structure. The structure is built in three stages: raw messages are divided into local *segments*; each segment is then *decoupled* into memory components that isolate reusable facts, constraints, and updates; and related components are finally *aggregated* into higher-level groups for efficient access. Retrieval proceeds top-down over this structure, starting from relevant groups and components, and expanding to segments and original messages only when finer-grained textual evidence is needed.

3.1 Building a Revisable Memory Structure

We next describe how xMemory constructs the hierarchy and updates it as new memory arrives.

From messages to components. We first partition the message stream into contiguous *segments*, each capturing a coherent local event, because answer-critical evidence in agent memory is often expressed across neighbouring turns rather than in isolated messages. Segments preserve this intact local context, but they are still too coarse to retrieve directly. We therefore extract one or more *memory components* from each segment, where each component represents a reusable fact, constraint, attribute, relation, or state update. This is the *decoupling* step of xMemory: it separates the decisive evidence from the surrounding local context before retrieval, while retaining links back to the source segment so that intact context can still be recovered when needed.

From components to groups. Once evidence has been decoupled into memory components, xMemory organises these components into higher-level *groups*, which serve as compact access units for retrieval. This is the *aggregation* step: rather than building the hierarchy directly over raw spans or summaries, we aggregate over already identified evidence units. The role of grouping is to make high-level retrieval both efficient and discriminative. If groups are too broad, retrieval becomes less discriminative and more redundant; if they are too fragmented, related evidence is scattered across many small units, weakening coverage for multi-fact queries. We therefore organise components into groups using an objective that balances *sparsity* and *semantic faithfulness*.

Let $\mathcal{P} = \{C_k\}_{k=1}^{K}$ denote a partition of N memory components into K groups, where $n_k = |C_k|$. We score an organisation by

$$f(\mathcal{P}) = \mathrm{SparsityScore}(\mathcal{P}) + \mathrm{SemScore}(\mathcal{P}). \tag{1}$$

The sparsity term favours balanced groups by using the normalised inverse of the expected within-group scanning cost:

$$\mathrm{SparsityScore}(\mathcal{P}) = \frac{N^2}{K \sum_{k=1}^{K} n_k^2}. \tag{2}$$

The semantic term encourages intra-group coherence and keeps the inter-group geometry navigable by Stage I's kNN-based retrieval (Eq. (4)).

$$\mathrm{SemScore}(\mathcal{P}) = \frac{1}{K} \sum_{k=1}^{K} \left(\frac{1}{n_k} \sum_{i \in C_k} \cos(\mathbf{x}_i, \boldsymbol{\mu}_k) \right) \cdot g(s_k), \tag{3}$$

where $\mathbf{x}_i$ is the embedding of component i, $\boldsymbol{\mu}_k$ is the centroid of group k, and $g(s_k) = \exp\!\left(-\frac{(s_k - \bar{s})^2}{2\sigma^2}\right)$, with $s_k = \max_{j \neq k} \cos(\boldsymbol{\mu}_k, \boldsymbol{\mu}_j)$, $\bar{s} = \mathrm{median}(\{s_k\})$, and $\sigma = \mathrm{median}(\{|s_k - \bar{s}|\}) + \varepsilon$. The bell-shaped $g(s_k)$ acts on inter-group geometry: near-duplicate centroids (s_k above the median) reintroduce the redundancy that decoupling removes, while isolated ones (s_k below the median) drop out of Stage I's kNN expansion as "semantic islands" [18]. Distinctiveness of individual components is already preserved upstream by decoupling; $g(s_k)$ only shapes group-centroid geometry.

Dynamic maintenance. Because agent memory evolves over time, the memory structure is maintained incrementally rather than fixed after construction. A new component with embedding $\mathbf{x}$ is attached to its nearest group $k^\star = \arg\max_{1 \leq k \leq K} \cos(\mathbf{x}, \boldsymbol{\mu}_k)$ if the similarity exceeds a threshold; otherwise a new group is created. Groups that become too large or internally incoherent are considered for splitting, while small or isolated groups are considered for merging. In both cases, the selected operation is the one that most improves Eq. (1).

We additionally maintain top-k nearest-neighbour links among groups to support local retrieval. Further details and a qualitative example are provided in Appendices A.1, A.2, and D.

3.2 Retrieval from the Memory Structure

The memory structure separates two roles that raw interaction history entangles: groups and memory components expose high-level evidence, while segments and original messages preserve the intact

local context in which that evidence was expressed. Retrieval therefore proceeds in two stages. We first identify a compact high-level backbone of relevant evidence, and only then expand to lower-level text when more context is necessary for answer generation.

Stage I: high-level backbone selection. Answers in agent memory are often supported by multiple related evidence units rather than a single best match. Stage I therefore selects a compact set of complementary high-level units, rather than repeatedly retrieving near-duplicate evidence by pure similarity ranking. To support this, we maintain kNN links among groups in the memory structure. These links record which high-level units are semantically close, even when they belong to different groups, allowing Stage I to avoid repeatedly selecting near-duplicate evidence and to identify related evidence regions that remain uncovered.

We first retrieve a small pool of groups by similarity between query and centroid, and include their neighbouring groups and associated components as candidate set V. Let $R \subseteq V$ denote the selected evidence backbone. For each candidate node $i \in V$, let $\mathcal{N}(i)$ denote its neighbours under the kNN links, and let $w_{iu} > 0$ denote the similarity weight on edge (i, u). We define the covered set as $C(R) = \{u \in V \mid \exists r \in R,\ u \in \{r\} \cup \mathcal{N}(r)\}$, and the newly covered nodes contributed by candidate i as $\Delta(i; R) = (\{i\} \cup \mathcal{N}(i)) \setminus C(R)$. At each step, we greedily select the next unit by trading off structural coverage against query relevance:

$$i^\star = \arg\max_{i \in V \setminus R} \left[\frac{\sum_{u \in \Delta(i;R)} w_{iu}}{Z} + \tilde{s}(q, i) \right], \tag{4}$$

where $\tilde{s}(q, i) \in [0, 1]$ is the normalised query–node similarity, and Z denotes the total candidate coverage weight. We apply Eq. (4) hierarchically, first selecting groups and then refining to components.

Stage II: adaptive text expansion. The high-level backbone indicates where useful evidence resides, but it does not yet determine how much original text should be revealed to the reader. Expanding all linked segments and messages would often reintroduce the same redundancy that the high-level structure is designed to avoid. Stage II therefore adds lower-level text only when it further reduces the reader's uncertainty.

Starting from the selected components, we gather their linked segments and construct a coarse context. Let $U(C, q)$ denote the reader's uncertainty under context C and query q, instantiated in our implementation via predictive entropy; concrete definitions and implementation details are given in Appendix A.3. For a candidate segment s, its marginal uncertainty reduction is

$$\Delta U(s \mid C, q) = U(C, q) - U(C \cup \{s\}, q). \tag{5}$$

A segment is included only if it yields a positive marginal reduction, i.e., $\Delta U(s \mid C, q) > 0$. For a candidate message m, we analogously compute $\Delta U(m \mid C, q) = U(C, q) - U(C \cup \{m\}, q)$ and include it only when it further reduces uncertainty. Retrieval stops when no remaining candidate segment or message provides additional uncertainty reduction.

4 Experiments

4.1 Experimental Setup

Datasets and metrics. We evaluate long-term agent memory on two complementary benchmarks: LoCoMo [19] and PerLTQA [20]. LoCoMo contains 50 multi-session dialogues, with an average of $\sim$18K tokens and $\sim$300 turns. Following prior work, we report results on its four answerable categories and omit the adversarial subset, which does not provide gold answers for the metrics used here. PerLTQA evaluates personalised long-term memory over longer contexts (around 25K tokens on average), with answers that are often sentence-style rather than short spans. We report BLEU-1 [21] and token-level F1 on both datasets, and additionally ROUGE-L [22] on PerLTQA to better capture sequence-level overlap for longer-form answers. As a complementary semantics check, we also report GPT-5-mini-judged answer accuracy in Appendix C.6.

Baselines. We compare against six baselines, grouped by how they access and organise memory. These baselines cover full-context reading, flat similarity retrieval, post-retrieval compression, and

Table 1: Main results on LoCoMo. Methods marked with [F] are flat retrieval baselines, and methods marked with [S] are structured memory baselines. We report BLEU and F1 for each question category and the average. Token/query denotes the average total tokens per query (lower is better) during inference time. Best results within each backbone model are in bold.

Model	Method	Multi-hop		Temporal		Open-domain		Single-hop		Average		Token /query
		BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	
Qwen3-8B	Full Memory[F]	24.63	31.66	10.02	12.58	12.48	17.59	33.80	44.00	25.83	33.54	18535.90
	Naive RAG[F]	22.46	34.28	17.21	21.32	12.35	17.08	35.68	45.22	27.95	36.48	8633.28
	LightMem[F]	19.63	26.23	22.67	27.61	9.83	14.37	31.34	40.66	26.04	33.66	5545.35
	Nemori[S]	24.68	36.82	25.78	33.76	12.71	18.48	38.02	47.52	31.44	40.88	7754.66
	A-Mem[S]	23.32	33.26	22.29	32.53	9.08	16.72	33.61	42.52	27.84	37.13	9103.46
	MemoryOS[S]	17.12	21.74	26.84	32.26	14.53	16.25	35.83	40.37	29.20	33.76	7234.66
	xMemory **(Ours)**	**27.24**	**38.57**	**29.58**	**37.46**	**15.55**	**20.69**	**40.94**	**50.94**	**34.48**	**43.98**	**4711.29**
Llama-3.1-8B-Ins	Full Memory[F]	15.11	21.52	6.06	8.06	7.78	11.13	23.29	35.13	17.23	25.50	18524.70
	Naive RAG[F]	13.77	19.40	6.20	8.61	6.99	10.22	21.90	35.37	16.21	25.30	11522.60
	LightMem[F]	16.43	23.76	12.44	15.61	9.83	13.70	21.68	25.08	18.05	22.15	5708.11
	Nemori[S]	18.18	26.30	19.23	26.09	9.24	12.06	26.19	40.18	22.21	32.95	9802.69
	A-Mem[S]	15.88	20.90	19.92	24.91	9.92	11.34	24.86	39.90	21.52	31.51	10268.77
	MemoryOS[S]	13.26	17.83	19.88	23.95	11.48	12.92	24.76	28.67	20.81	24.72	7212.07
	xMemory **(Ours)**	**22.21**	**30.99**	**21.20**	**27.42**	**11.58**	**14.61**	**28.43**	**41.15**	**24.73**	**34.77**	**5539.97**
GPT-5 nano	Full Memory[F]	22.95	34.07	20.02	23.50	21.61	26.42	32.71	46.97	27.58	38.44	18544.25
	Naive RAG[F]	21.34	32.31	28.43	29.05	22.63	26.56	36.68	46.35	31.28	38.94	7531.46
	LightMem[F]	23.13	32.07	41.06	55.23	21.27	26.38	37.52	44.31	34.60	43.23	6850.04
	Nemori[S]	24.80	37.11	41.56	54.25	22.61	29.29	40.35	51.67	36.65	48.17	9154.76
	A-Mem[S]	24.12	35.88	40.87	54.67	20.83	27.61	38.42	46.13	35.22	44.88	9610.94
	MemoryOS[S]	24.47	36.13	39.78	55.34	21.45	28.71	39.39	48.68	35.62	46.53	7029.02
	xMemory **(Ours)**	**27.56**	**39.97**	**46.10**	**57.62**	**25.53**	**30.92**	**41.14**	**52.63**	**38.71**	**50.00**	**6581.20**

representative structured-memory designs. **Flat or minimally structured baselines** include **(1) Full Memory**, which provides the complete available history to the reader model without retrieval; **(2) Naive RAG**, which chunks original messages and retrieves the top-20 chunks by vector similarity; and **(3) LightMem**, which reduces retrieval cost through multi-stage filtering and compression [7]. **Structured memory baselines** include **(4) Nemori**, which builds a hierarchical memory from episodic memories to higher-level semantic memories for long-term recall [16]; **(5) A-Mem**, which stores memory as structured notes connected by dynamic links [9]; and **(6) MemoryOS**, which organises memory into temporally layered storage with lifecycle management [8].

Implementation details. We evaluate all compared methods with three reader LLMs: two recent open-source models, Qwen3-8B [23] and Llama-3.1-8B-Instruct [24], and one closed-source model, GPT-5 nano. By default, final answers are generated with greedy decoding (temperature $= 0.0$) for deterministic evaluation [25]. All retrieval and memory construction embeddings use `text-embedding-3-small` by default, and each method uses the same backbone model for memory construction and answer generation. We further evaluate sensitivity to the embedding model (BGE-M3) and decoding temperature ($T = 0.6$), with results reported in Appendix C.7. For GPT-5 nano, which does not expose token-level logits, we estimate uncertainty with GPT-4.1-mini. All xMemory hyperparameters are fixed across datasets and backbone models; their values and implementation details are reported in Appendix B for reproducibility.

4.2 Main Results

LoCoMo: retrieval over coherent multi-session histories. Table 1 reports results on LoCoMo. Across all three backbones, xMemory achieves the best average performance, with especially clear gains on multi-hop and temporal questions. For example, with Qwen3-8B, average BLEU/F1

Table 2: Main results on PerLTQA. Methods marked with [F] are flat retrieval baselines, and methods marked with [S] are structured memory baselines. We report BLEU, F1, R-L (ROUGE-L), and Tok. (token usage per query during inference time, lower is better). Best results within each backbone model are in bold.

Method	Qwen3-8B				Llama-3.1-8B-Ins				GPT-5 nano			
	BLEU	F1	R-L	Tok.	BLEU	F1	R-L	Tok.	BLEU	F1	R-L	Tok.
Full Memory[F]	32.73	42.15	36.65	25045	33.00	44.40	39.43	25098	24.37	34.97	29.81	25212
Naive RAG[F]	32.08	41.37	35.95	6274	33.67	44.84	39.65	9531	27.35	37.76	32.35	10756
LightMem[F]	29.12	40.21	34.67	7692	31.83	42.33	38.37	**5452**	24.84	34.51	29.33	7579
Nemori[S]	32.55	42.80	38.05	9092	41.01	49.62	44.65	11440	33.44	41.79	38.43	11883
A-Mem[S]	31.36	40.92	36.45	9864	35.56	45.79	41.38	7707	33.12	41.17	37.94	14718
MemoryOS[S]	35.14	42.35	38.48	6499	34.79	42.03	38.22	6511	27.66	33.83	31.44	12669
xMemory **(Ours)**	**36.24**	**47.08**	**42.50**	**5087**	**42.68**	**52.37**	**47.84**	6066	**36.79**	**46.23**	**41.25**	**7307**

improves from 31.44/40.88 for Nemori to **34.48/43.98**; with GPT-5 nano, it improves from 36.65/48.17 to **38.71/50.00** while reducing token usage from 9155 to **6581** per query.

Compared with both flat retrieval and structured memory baselines, these results suggest that agent memory benefits from decoupling decisive evidence before high-level organisation and retrieval. Full Memory and Naive RAG preserve large amounts of raw context but often return redundant or weakly discriminative evidence, while LightMem reduces cost mainly through post-retrieval compression and performs worse on multi-hop and temporal questions. Structured baselines introduce higher-level organisation, but their memory structures are not explicitly centred on decoupled evidence units and revisable grouping. By contrast, xMemory first decouples distinctive evidence into memory components, organises them into revisable groups, and retrieves top-down over this structure.

PerLTQA: generalisation to longer personalised memory. Table 2 reports results on PerLTQA, which contains longer contexts and more sentence-style answers than LoCoMo. xMemory remains consistently effective across all three backbone models, showing that the proposed retrieval principle transfers beyond multi-session dialogue recall to longer personalised memory reasoning. With Qwen3-8B, it achieves the best BLEU/F1/ROUGE-L at **36.24/47.08/42.50** while using the fewest tokens; with Llama-3.1-8B-Instruct, it reaches **42.68/52.37/47.84**. These gains suggest that xMemory improves selective evidence access while preserving the information needed for coherent sentence-level reconstruction. The semantic accuracy confirms this trend: xMemory leads in all six dataset–reader settings by 1.81–4.02 percentage points over the strongest alternative (Appendix C.6).

Efficiency. We report *tokens per query* as the average end-to-end inference cost, including retrieval, answer generation, and auxiliary calls. Across both datasets, xMemory achieves better answer quality with competitive or fewer tokens than strong memory baselines, indicating that its gains come from delivering more concentrated evidence rather than exposing more history. For example, on LoCoMo with Qwen3-8B, xMemory reduces token usage from 7755 for Nemori to **4711** while also improving average BLEU/F1 from 31.44/40.88 to **34.48/43.98**. These results suggest that token efficiency in agent memory should be measured by evidence utility rather than context reduction alone.

5 Analysis

5.1 Ablation Studies

Retrieval stage analysis. We first ablate memory structuring and the two retrieval stages on LoCoMo with Qwen3-8B. As shown in Figure 3 (a), Memory-only improves average BLEU/F1 from 27.95/36.48 for Naive RAG to 31.81/40.77, showing that retrieval over decoupled memory units is more effective than retrieval over flat raw chunks, even with basic similarity matching. Adding Stage I improves selection over groups and components, while Stage II improves uncertainty guided expansion to segments and messages. Combining both stages gives the best trade-off, with the highest average BLEU/F1 and the lowest token usage.

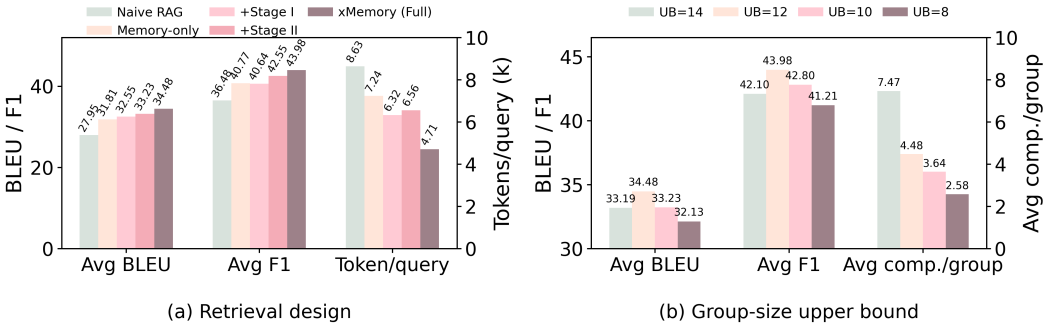

Figure 3: **Ablation studies on LoCoMo with Qwen3-8B.** (a) Retrieval-stage ablation summarising the effect of memory structuring, Stage I, and Stage II on average BLEU/F1 and token usage. (b) Effect of the upper bound (UB) of memory components per group, showing average BLEU/F1 and the resulting average number of components per group.

Group size upper bound analysis. We next study the upper bound on the number of memory components per group. This parameter controls routing arity: a larger candidate set within each group makes decisive evidence harder to identify in dialogue memory with high semantic similarity, while an overly small group size scatters related facts. Guided by a Fano style lower bound, which shows that routing error increases with candidate size when discriminative information is bounded, we choose 12 as a practical threshold and then validate it empirically. Figure 3 (b) shows that this setting performs best, reaching 34.48 BLEU and 43.98 F1 with 4.48 components per group on average. Larger groups weaken discrimination, while smaller groups fragment the hierarchy. Full results and the detailed theoretical motivation are provided in Appendix C.1 and Appendix C.2.

5.2 Retroactive Restructuring in Memory Construction

Unlike standard RAG, where the retrieval corpus is usually fixed, agent memory evolves as new interactions arrive. In xMemory, later insertions can trigger *split* or *merge* operations over high-level groups, thereby revising the assignment of previously created memory components. We measure this effect with the **dynamic reassignment ratio**, defined as the fraction of existing components whose group assignment changes during later insertions.

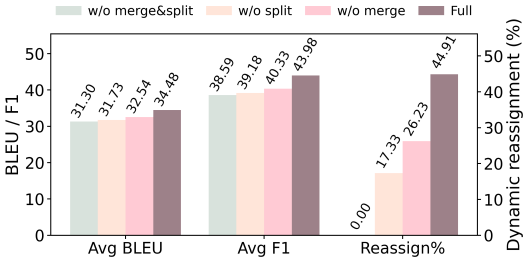

Figure 4: **Retroactive restructuring.** Freezing high-level restructuring yields 0% reassignment and lower QA performance, while the full system performs substantial reassignment and achieves the best Avg. BLEU/F1.

Figure 4 shows that such retroactive restructuring improves downstream QA. Disabling both *split* and *merge* freezes the structure, yields 0% reassignment, and reduces average F1 to 38.59, whereas the full system reaches the highest reassignment ratio (44.91%) and the best average F1 (43.98). The structure statistics in Appendix C.4 further show that *split* and *merge* play complementary roles: *split* enables revision by repartitioning broad groups, while *merge* prevents the high-level index from becoming unnecessarily fragmented. These results support our claim that revisability is not merely an implementation detail, but a useful property for memory organisation under evolving evidence.

5.3 Retrieval Efficiency and Cost Performance Trade-off

Evidence density and coverage efficiency. To understand why xMemory improves answer quality while reducing inference-time context cost, we compare three retrieval settings on LoCoMo with Qwen3-8B: (1) Naive RAG with top-20 chunks; (2) RAG with the LLMLingua-2 pruning module [26] as in LightMem [7]; and (3) xMemory. For each query, we remove stopwords from the reference answer and treat the remaining content words as answer evidence units. A retrieved block refers

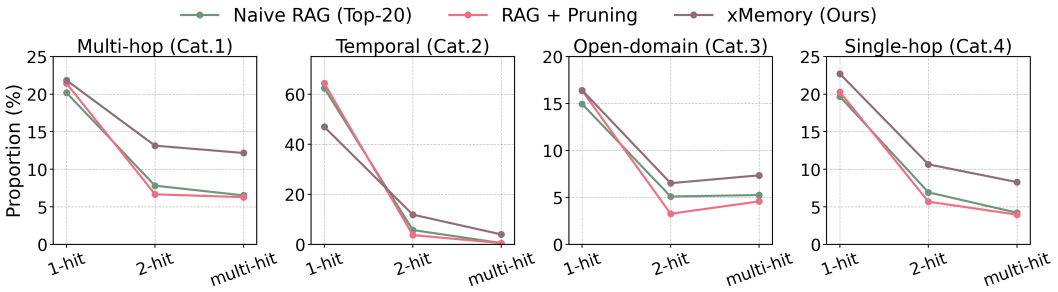

Figure 5: Evidence hit distribution by question category. Each subplot shows the proportions of retrieved blocks that are 1-hit, 2-hit, or multi-hit, comparing Naive RAG, RAG with pruning, and xMemory. Higher 2-hit and multi-hit proportions indicate denser evidence retrieval.

to a raw/pruned chunk for RAG, or a segment or selected component set for xMemory, and is classified as **1-hit**, **2-hit**, or **multi-hit** based on whether it contains one, two, or at least three distinct evidence units. Figure 5 shows that xMemory retrieves denser evidence across all question categories, with consistently higher 2-hit and multi-hit proportions than both RAG baselines. Token cost is measured by the actual text passed to the reader. This suggests that decoupling memory into finer-grained components helps concentrate answer-relevant evidence before retrieval, rather than leaving it dispersed across redundant raw chunks. By contrast, pruning shifts mass from 2-hit and multi-hit blocks toward 1-hit blocks, suggesting that post-retrieval compression can fragment entangled evidence even when the remaining context is broadly relevant. For example, on multi-hop questions, xMemory increases the 2-hit and multi-hit proportions to 13.14% and 12.19%, compared with 7.82% and 6.53% for Naive RAG.

We further evaluate *coverage efficiency* by comparing how many retrieved blocks and tokens are needed to cover all answer evidence units. Table 3 shows that pruning reduces token cost relative to Naive RAG, but requires more blocks and yields lower accuracy, consistent with fragmented evidence after compression. In contrast, xMemory achieves the best BLEU/F1 while covering answer evidence with substantially fewer blocks and tokens. This suggests that xMemory improves not simply by retrieving less context, but by concentrating more answer-relevant evidence into a smaller retrieval budget.

Table 3: **Average evidence coverage efficiency on LoCoMo.** Blocks denotes the average number of retrieved blocks needed to cover all answer evidence units, and Tokens denotes the corresponding token cost. Full category-level results are in Appendix C.3.

Method	BLEU	F1	Blocks↓	Tokens↓
Naive RAG	27.95	36.48	10.81	1979.26
RAG + Pruning	26.55	34.58	13.31	1587.99
xMemory	**34.48**	**43.98**	**5.66**	**974.56**

Table 4: **Sensitivity to coverage weight α.** Results on LoCoMo (F1) and AMA-Bench (GPT-5-mini-judged accuracy, %) with Qwen3-8B. Best results are in bold.

	LoCoMo (F1)					AMA-Bench (Accuracy)				
α	Multi	Temp.	Open	Single	Avg.	Recall	Causal	Update	Abstr.	W. Avg.
0.3	37.63	36.21	19.39	48.67	42.23	39.21	37.30	28.85	25.47	33.79
0.5	38.57	37.46	**20.69**	50.94	43.98	**39.69**	**40.77**	**31.99**	**28.26**	**36.06**
0.7	**39.27**	**37.52**	20.58	**52.21**	**44.80**	35.76	39.88	30.12	25.47	33.57
0.9	34.42	33.02	18.52	48.35	40.74	35.32	38.72	30.23	24.68	33.04

Balancing coverage and relevance. We analyze the balance between neighborhood coverage and query relevance in Stage I by weighting the two terms in Eq. (4) with α and $1-\alpha$, respectively, where $\alpha = 0.5$ corresponds to the default equal-weight formulation. As shown in Table 4, LoCoMo achieves its highest F1 at $\alpha = 0.7$ (44.80), while AMA-Bench performs best at $\alpha = 0.5$ (36.06%). This suggests that dialogue memories with diverse topics may benefit from stronger coverage, whereas task-centered agent trajectories require greater emphasis on query relevance. Increasing α to

0.9 degrades performance on both benchmarks, indicating that excessive coverage weighting may overlook answer-critical information.

Overall cost and performance. The above analysis focuses on inference-time retrieval efficiency. We next examine whether this advantage remains after accounting for memory construction cost. Following the LightMem-style reporting setup [7], we calculate construction tokens and compute total token cost as average construction cost plus average inference tokens per query multiplied by the corresponding query count. The exact statistics are reported in Appendix C.5.

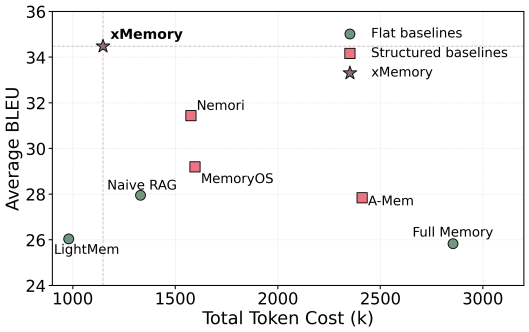

Figure 6: **Cost–performance trade-off on LoCoMo.** Total token cost is plotted against average BLEU with Qwen3-8B.

As shown in Figure 6, xMemory achieves the most favourable trade-off. Flat baselines such as Naive RAG and LightMem use relatively low or moderate token budgets, but obtain lower answer quality. Structured memory baselines improve over flat retrieval in some cases, but usually incur higher construction and inference costs. In contrast, xMemory attains the highest average BLEU while using a lower total token cost than Nemori, MemoryOS, and A-Mem. This suggests that the denser evidence retrieved by xMemory not only improves local coverage efficiency, but also translates into a better end-to-end efficiency–quality balance after amortising construction cost.

Storage overhead and answer quality. To examine the storage–quality trade-off of structured memory methods, we extend our analysis beyond LoCoMo and PerLTQA to two additional benchmarks: LongMemEval (∼115k tokens), which evaluates long-term multi-session memory, and AMA-Bench (∼57k tokens), which covers agent action–observation trajectories. For LongMemEval, we retain the full memory streams and evaluate a category stratified sample of 102 questions across methods.

Table 5: **Storage–accuracy trade-off.** On-disk memory storage (MB) / GPT-5-mini-judged accuracy (%), using Qwen3-8B for construction and answering.

Method	LoCoMo	PerLTQA	LongMemEval	AMA
A-Mem	**1.41**/45.19	**1.04**/42.80	**1.21**/39.22	0.50/27.28
MemoryOS	12.24/47.40	5.83/45.60	10.53/37.25	2.52/29.29
Nemori	5.89/47.99	6.25/46.20	5.21/41.18	1.46/31.69
xMemory	1.57/**52.01**	1.50/**49.70**	1.44/**47.06**	**0.28/36.06**

As shown in Table 5, xMemory achieves the highest semantic accuracy across all four benchmarks, outperforming the strongest structured baseline by 3.50–5.88 percentage points. Despite its hierarchical design, xMemory maintains a compact storage footprint (0.28–1.57 MB), comparable to A-Mem and consistently smaller than MemoryOS and Nemori. Notably, its footprint does not increase proportionally with raw context length, suggesting that memory diversity and component reuse also influence storage requirements. Together with the amortized token costs in Appendix C.5, these results demonstrate a favorable storage–quality and processing-cost trade-off.

6 Conclusion

We presented xMemory, a retrieval framework that uses hierarchical organisation to address the mismatch between agent memory and standard RAG assumptions. By disentangling highly correlated memories into semantic components and retrieving top-down over the resulting hierarchy, xMemory reduces redundancy while preserving temporally linked evidence. Across LoCoMo and PerLTQA, xMemory improves answer quality with lower token cost and retrieves more evidence-dense contexts than RAG baselines. These results suggest that effective agent memory should not only retrieve or compress past interactions, but also reorganise them into evidence-oriented structures that make subtle distinctions easier to access. We hope xMemory provides a useful step toward more adaptive and efficient memory systems for long-horizon LLM agents.

Acknowledgments

This work was supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) through a Turing AI Fellowship (grant no. EP/V020579/1 and EP/V020579/2), the Prosperity Partnership scheme (grant no. UKRI566), and King's College London's Impact Acceleration Account (grant no. EP/X525571/1). The authors also acknowledge the use of the King's Computational Research, Engineering, and Technology Environment (CREATE) at King's College London.

References

[1] Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Sean Follmer, Jeff Han, Jürgen Steimle, and Nathalie Henry Riche, editors, *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pages 2:1–2:22. ACM, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

[2] Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8416–8439, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.413. URL https://aclanthology.org/2025.acl-long.413/.

[3] Tiannan Wang, Meiling Tao, Ruoyu Fang, Huilin Wang, Shuai Wang, Yuchen Eleanor Jiang, and Wangchunshu Zhou. AI PERSONA: towards life-long personalization of llms. *CoRR*, abs/2412.13103, 2024. doi: 10.48550/ARXIV.2412.13103. URL https://doi.org/10.48550/arXiv.2412.13103.

[4] Yexiong Lin, Shanshan Ye, Yu Yao, Zhen Fang, Bo Han, and Tongliang Liu. Squidagent: Parallelize wisely, coordinate efficiently. *arXiv preprint arXiv:2610.08647*, 2026.

[5] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 19724–19731. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

[6] Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. doi: 10.48550/ARXIV.2310.08560. URL https://doi.org/10.48550/arXiv.2310.08560.

[7] 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.

[8] Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25961–25970, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

[9] Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

[10] Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.

[11] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In

H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

[12] Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. *CoRR*, abs/2501.13956, 2025. doi: 10.48550/ARXIV.2501.13956. URL https://doi.org/10.48550/arXiv.2501.13956.

[13] Dongming Jiang, Yi Li, Songtao Wei, Jinxin Yang, Ayushi Kishore, Alysa Zhao, Dingyi Kang, Xu Hu, Feng Chen, Qiannan Li, and Bingzhe Li. Anatomy of agentic memory: Taxonomy and empirical analysis of evaluation and system limitations. *CoRR*, abs/2602.19320, 2026. doi: 10.48550/ARXIV.2602.19320. URL https://doi.org/10.48550/arXiv.2602.19320.

[14] Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the age of AI agents. *CoRR*, abs/2512.13564, 2025. doi: 10.48550/ARXIV.2512.13564. URL https://doi.org/10.48550/arXiv.2512.13564.

[15] Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, Shengyuan Chen, Huachi Zhou, Qinggang Zhang, Ninghao Liu, Jinsong Su, Xinrun Wang, Yi Chang, and Xiao Huang. Graph-based agent memory: Taxonomy, techniques, and applications. *CoRR*, abs/2602.05665, 2026. doi: 10.48550/ARXIV.2602.05665. URL https://doi.org/10.48550/arXiv.2602.05665.

[16] Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. Nemori: Self-organizing agent memory inspired by cognitive science. *CoRR*, abs/2508.03341, 2025. doi: 10.48550/ARXIV.2508.03341. URL https://doi.org/10.48550/arXiv.2508.03341.

[17] Rui Li, Zeyu Zhang, Xiaohe Bo, Zihang Tian, Xu Chen, Quanyu Dai, Zhenhua Dong, and Ruiming Tang. CAM: A constructivist view of agentic memory for LLM-based reading comprehension. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=ACSOnSHiWe.

[18] Yaoze Zhang, Rong Wu, Pinlong Cai, Xiaoman Wang, Guohang Yan, Song Mao, Ding Wang, and Botian Shi. Leanrag: Knowledge-graph-based generation with semantic aggregation and hierarchical retrieval. *arXiv preprint arXiv:2508.10391*, 2025.

[19] Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.747. URL https://doi.org/10.18653/v1/2024.acl-long.747.

[20] Yiming Du, Hongru Wang, Zhengyi Zhao, Bin Liang, Baojun Wang, Wanjun Zhong, Zezhong Wang, and Kam-Fai Wong. PerLTQA: A personal long-term memory dataset for memory classification, retrieval, and fusion in question answering. In Kam-Fai Wong, Min Zhang, Ruifeng Xu, Jing Li, Zhongyu Wei, Lin Gui, Bin Liang, and Runcong Zhao, editors, *Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10)*, pages 152–164, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.sighan-1.18/.

[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages

311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

[22] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013/.

[23] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jian Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

[24] Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

[25] 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 Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14918–14937. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.923. URL https://doi.org/10.18653/v1/2023.emnlp-main.923.

[26] Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.57. URL https://aclanthology.org/2024.findings-acl.57/.

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

[28] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In Inês Lynce, Nello Murano, Mauro Vallati, Serena Villata, Federico Chesani, Michela Milano, Andrea Omicini, and Mehdi Dastani, editors, *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pages 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Appendix

A Additional Method Details 16
A.1 From Raw Messages to Segments and Components 16
A.2 Detailed Structure Maintenance 16
A.3 Detailed Retrieval Implementation 17
A.4 Pseudo-code of the Full Pipeline 18
B Implementation Setup 18
C More Experiment Results 19
C.1 Detailed Ablation Results 19
C.2 Theoretical Motivation of Branching Factor 19
C.3 Evidence Density under RAG and Pruning Effect 22
C.4 Retroactive Restructuring in Memory Construction 22
C.5 Additional Cost–Performance Statistics 22
C.6 Semantic Answer Evaluation 23
C.7 Sensitivity to Embedding and Decoding Settings 23
D Case Study: Decoupling before Aggregation 24
E Limitations 26
F Broader Impact Statement 26
G Prompt Design 26

A Additional Method Details

This appendix provides implementation details omitted from the main text, including segment construction, memory component extraction, dynamic structure maintenance, and retrieval-time uncertainty estimation.

A.1 From Raw Messages to Segments and Components

Segment construction. The interaction history is first partitioned into contiguous *segments*, each intended to capture a coherent local event. In practice, segmentation is performed incrementally over the message stream. A new incoming message is attached to the current segment when it remains part of the same local interaction event, and starts a new segment otherwise. This decision is based on local semantic continuity, including topical consistency, temporal continuity, and explicit discourse shifts. The resulting segments preserve intact contextual evidence that may later be needed for reconstruction.

Memory component extraction. Given a segment, xMemory uses an LLM to extract one or more *memory components*. Each component is a concise structured unit representing a reusable fact, constraint, attribute, preference, relation, or state update expressed in the segment. The extraction prompt instructs the LLM to decompose a segment into minimal evidence units that are likely to be useful for future question answering, while avoiding unnecessary duplication across components from the same segment.

Each component is stored together with (1) its textual description, (2) an embedding for retrieval and grouping, and (3) a pointer to its source segment. This design allows the memory structure to operate over fine-grained evidence units while retaining access to the original local context.

Component granularity. We aim for components that are finer-grained than segments but still semantically self-contained. In particular, a component should isolate a single reusable piece of evidence whenever possible, rather than conflate multiple loosely related facts. This granularity is important because highly similar segments often differ only in a small but answer-critical detail. Extracting such details as separate components makes them directly retrievable at the high level.

A.2 Detailed Structure Maintenance

The main text introduces a guidance objective for organising components into groups and states that the memory structure is maintained incrementally. We provide more details here.

Attach operation. Suppose a new memory component with embedding $\mathbf{x}$ arrives. Let $\boldsymbol{\mu}_k$ denote the centroid embedding of group k. We first identify the nearest group

$$k^\star = \arg \max_{1 \le k \le K} \cos(\mathbf{x}, \boldsymbol{\mu}_k).$$

If the similarity to this group exceeds a threshold τ_{attach}, the component is attached to group $k^\star$; otherwise a new group is created. The threshold controls whether the current organisation is considered sufficiently expressive for the incoming evidence.

Split triggers. A group is considered for splitting when it becomes too broad to serve as a useful high-level access unit. In our implementation, this happens when the group size exceeds a predefined threshold. We generate candidate partitions using graph-based local clustering on component embeddings: components are connected if their pairwise similarity exceeds a fixed threshold, and the resulting connected components form candidate clusters. A size-based fallback partition is used when necessary. Among these candidate partitions, we select the split that yields the largest improvement in Eq. (1).

Merge triggers. A group is considered for merging when it contains only a single memory component and is therefore unlikely to provide a stable high-level access unit on its own. Candidate merge targets are drawn from its nearest neighbouring groups in centroid space, and the merge with the largest improvement in Eq. (1) is selected.

Maintenance schedule. The attach operation is applied online whenever a new component is created. Split and merge are applied periodically, or when the arrival of new components causes local structural statistics to cross the corresponding thresholds. This makes the memory structure revisable rather than fixed: later observations can reorganise earlier evidence when better relations become apparent.

kNN link updates. In addition to group membership, xMemory maintains top-k nearest-neighbour links among groups and components in embedding space. These links are updated whenever new components are added and whenever split or merge operations alter the local structure. To avoid global recomputation after every update, kNN maintenance can be restricted to the affected neighbourhood.

Complexity. The component-to-group organisation keeps high-level retrieval efficient by reducing the number of units examined at coarse stages. Dynamic maintenance introduces additional cost, but this cost is amortised over memory updates and is confined to local structural revisions rather than full reconstruction of the memory structure. In practice, attach is inexpensive, while split and merge are triggered much less frequently.

A.3 Detailed Retrieval Implementation

This section provides details omitted from the main text for candidate generation, backbone selection, and uncertainty-based expansion.

Initial candidate generation. Given a query q, we first embed the query and retrieve a candidate set of groups and memory components from the memory structure. Candidate generation is performed independently at the group level and the component level using embedding similarity. The union of these candidates forms the high-level candidate set V used in Stage I. This design allows retrieval to enter the search space both through coarse semantic regions and through directly matched fine-grained evidence units.

Backbone selection. Stage I selects a compact set of complementary high-level units by balancing query relevance and structural coverage, as defined in Eq. (4). In practice, the procedure is greedy. At each step, the system recomputes the marginal gain of each remaining candidate under the current selected set R, adds the best-scoring candidate, and terminates when the candidate set is exhausted.

The coverage term is defined over local kNN neighbourhoods rather than the full graph, which keeps the computation sparse and prevents the selected set from collapsing into a single dense region of near-duplicate evidence.

Hierarchy-aware refinement. Although Eq. (4) is written over a generic candidate set, we apply it hierarchically in practice. We first select a compact set of groups, then restrict attention to the components associated with those groups, and finally apply the same selection principle again to obtain the component-level evidence backbone. This reduces search cost and encourages diversity at both levels.

Segment expansion. Each selected memory component points back to its source segment. Stage II first identifies these linked segments as candidates for recovering local event context. Each candidate is evaluated by its marginal uncertainty reduction under the current context. Full-segment text is included only when it provides additional information useful for answering the query.

Message expansion. Messages provide a finer-grained alternative to full-segment expansion. When the useful evidence is concentrated in specific utterances, Stage II evaluates constituent messages individually using the same uncertainty-reduction criterion. This allows the system to recover precise original evidence without unnecessarily including the entire segment. Already included text is not duplicated in the final retrieved context.

Uncertainty estimation. The main text uses $U(C, q)$ to denote the reader's uncertainty under context C and query q. In our implementation, when the reader model exposes token-level predictive distributions, we instantiate $U(C, q)$ as the entropy of the next-token answer distribution under the

current context. This provides a scalar proxy for how uncertain the reader remains about the answer after seeing C.

When the reader model does not expose token-level predictive uncertainty, we estimate $U(\cdot)$ using a proxy model that provides token-level logits. The proxy model receives the same query and context, and its predictive entropy is used only for expansion decisions; the final answer is still generated by the designated reader model. This separation allows the retrieval policy to remain uncertainty-aware even for black-box or API-based readers.

Stopping rule. Stage II terminates when no remaining candidate segment or message yields positive marginal uncertainty reduction. This prevents unnecessary expansion into redundant raw text after the reader's uncertainty can no longer be further reduced by available candidates.

A.4 Pseudo-code of the Full Pipeline

For completeness, we summarise the full pipeline below.

Algorithm 1: xMemory

Input: Interaction history $H = \{m_1, \ldots, m_T\}$, query q, token budget B
Output: Retrieved context C
Partition H into segments;
Extract memory components from each segment;
Organise components into groups and build kNN links;
foreach *new component arrival* **do**
  Attach to nearest compatible group or create a new group;
  if *a group becomes too large or incoherent* **then**
    Split using Eq. (1);
  if *a group becomes too small or isolated, such as only including one memory component* **then**
    Merge using Eq. (1);
Retrieve candidate groups and components for query q;
Select a compact high-level evidence backbone using Eq. (4);
Initialise coarse context C from selected groups and components;
Gather candidate linked segments;
foreach *candidate segment s in ranked order* **do**
  if $\Delta U(s \mid C, q) > 0$ **then**
    Add s to C;
    foreach *message m in s* **do**
      if $\Delta U(m \mid C, q) > 0$ **then**
        Add m to C;
return C;

B Implementation Setup

xMemory hyperparameters. Unless otherwise specified, we use a group size cap of 12 memory components, an attachment threshold of 0.6, a post-split clustering threshold of 0.65, and a merge threshold of 0.65 across all datasets and backbone models. The attachment threshold controls whether a newly created component is assigned to an existing group, while the post-split clustering threshold controls the granularity of groups formed after a split operation. The merge threshold controls when a small or weakly separated group should be merged into a neighbouring group. Additionally, following prior structured retrieval [18, 27] and agent-memory systems [9], we set the kNN neighbourhood size to 10 and the Stage I candidate pool size to 20. Experiments with open-source models are run on an NVIDIA A100 80GB GPU.

Hyperparameter sensitivity. We evaluate the sensitivity of xMemory to the main structure-update thresholds on LoCoMo using Qwen3-8B for both memory construction and answer generation. For each experiment, we vary one threshold while keeping all other hyperparameters fixed. As shown in Table 6, the default attachment threshold of 0.6, post-split clustering threshold of 0.65, and merge

Table 6: Sensitivity analysis of the main structure-update thresholds on LoCoMo using Qwen3-8B. We vary one threshold at a time while keeping all other hyperparameters fixed. Results are reported as BLEU/F1, and the default values are marked in gray.

Threshold	Value	Multi-hop	Temporal	Open-domain	Single-hop	Average
Attachment	0.5	25.36 / 37.29	28.13 / 35.38	14.97 / 19.12	39.86 / 49.65	33.21 / 42.51
	0.6	27.24 / 38.57	29.58 / 37.46	15.55 / 20.69	40.94 / 50.94	34.48 / 43.98
	0.7	25.69 / 38.02	27.56 / 35.71	15.27 / 20.02	40.09 / 49.81	33.29 / 42.85
	0.8	24.95 / 36.66	27.35 / 35.65	14.20 / 18.43	39.56 / 49.42	32.76 / 42.28
Post-split clustering	0.55	26.82 / 37.91	28.82 / 36.78	14.59 / 20.06	39.10 / 49.20	33.18 / 42.73
	0.65	27.24 / 38.57	29.58 / 37.46	15.55 / 20.69	40.94 / 50.94	34.48 / 43.98
	0.75	26.80 / 38.22	28.85 / 36.27	14.06 / 19.79	39.86 / 50.51	33.57 / 43.37
Merge	0.6	26.34 / 37.21	28.65 / 36.51	14.88 / 19.40	39.70 / 49.21	33.40 / 42.51
	0.65	27.24 / 38.57	29.58 / 37.46	15.55 / 20.69	40.94 / 50.94	34.48 / 43.98
	0.7	26.05 / 37.10	28.23 / 35.78	15.19 / 19.98	39.34 / 48.80	33.08 / 42.14
	0.75	25.43 / 36.16	27.83 / 35.37	14.59 / 19.68	39.19 / 48.21	32.77 / 41.55

threshold of 0.65 achieve the best average BLEU and F1 in their respective sensitivity groups. The method remains relatively stable across nearby values, suggesting that its performance does not depend on a narrow threshold choice. Lower attachment thresholds may attach components too aggressively, while higher thresholds make the structure more conservative. Similarly, lower post-split thresholds may produce overly coarse groups, whereas higher thresholds may over-fragment related components. For merging, both lower and higher thresholds lead to weaker average performance, indicating that an intermediate value better preserves the balance between structural compactness and semantic separation.

Existing assets and licenses. We use existing datasets, models, and baseline methods only for research evaluation. LoCoMo and PerLTQA are credited through their original papers in Section 4.1. PerLTQA is released under the CC BY-NC 4.0 license for non-commercial research use. LoCoMo is released with its accompanying data and code repository; we follow the terms provided by the dataset authors. We use Qwen3-8B, Llama-3.1-8B-Instruct, GPT-5 nano, GPT-4.1-mini, and `text-embedding-3-small` according to their respective model licenses or API terms. Baseline methods are credited through their original papers, and any reused code in the released package follows the corresponding licenses and attribution requirements.

C More Experiment Results

C.1 Detailed Ablation Results

Table 7 reports the full category-level ablation results for the retrieval design on LoCoMo with Qwen3-8B. The results confirm that memory structuring alone already provides a substantially stronger retrieval basis than Naive RAG, and that both Stage I and Stage II contribute further improvements from complementary directions. Stage I mainly improves high-level evidence selection while reducing token usage, whereas Stage II contributes stronger gains in answer quality by uncertainty-guided expansion over segments and messages. The full system achieves the best overall trade-off across performance and inference-time token cost.

C.2 Theoretical Motivation of Branching Factor in Hierarchical Dialogue Summarization

C.2.1 Motivation and Fano Inequality

Our hierarchy performs top-down retrieval, in which key subroutines repeatedly *route* to a relevant unit by selecting from a candidate set (e.g., a group/memory component relevant to the current query context). In high-similarity dialogue streams, the observable signal used for routing (summaries/embeddings/LLM scores) can be inherently weak at discriminating among many near-duplicate candidates. This subsection summarises a standard information-theoretic implication:

Table 7: **Ablation on LoCoMo with Qwen3-8B.** We report BLEU, F1, and Token/query (lower is better). **Memory-only** uses the full memory structure but replaces both retrieval stages with basic similarity retrieval. **+Stage I** adds group and component selection. **+Stage II** adds uncertainty-based segment and message expansion. **Ours (Full)** combines both stages.

Setting	Multi-hop		Temporal		Open-domain		Single-hop		Average		Token
	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	/query
Naive RAG	22.46	34.28	17.21	21.32	12.35	17.08	35.68	45.22	27.95	36.48	8633.28
Memory-only	24.48	35.41	27.28	34.14	13.20	17.75	38.19	47.73	31.81	40.77	7235.56
+Stage I	26.55	38.02	28.08	35.72	14.30	19.75	38.36	45.79	32.55	40.64	6320.72
+Stage II	26.64	37.03	28.92	37.09	14.10	20.43	39.28	49.01	33.23	42.55	6556.58
xMemory (Full)	**27.24**	**38.57**	**29.58**	**37.46**	**15.55**	**20.69**	**40.94**	**50.94**	**34.48**	**43.98**	**4711.29**

with bounded discriminative information, the error of a multi-way routing decision cannot be made arbitrarily small as the candidate set grows.

Consider one such routing decision that must identify the correct option among n_k candidates within a group on the component–group level. Let $Z \in \{1, \dots, n_k\}$ be the (unknown) index of the correct candidate, and let O denote the observable evidence (e.g., the group summary derived from its constituent components and the query/context representations) used for routing. Any routing rule outputs $\hat{Z}(O)$ with error probability $p_e = \Pr[\hat{Z} \neq Z]$. A classical result (Fano's inequality) implies

$$p_e \;\geq\; 1 - \frac{I(Z;O) + 1}{\log_2 n_k}, \tag{6}$$

where $I(Z;O)$ denotes the mutual information between the routing evidence O (derived from the query/context and stored representations) and the correct candidate index Z. In our setting, candidates within the same group are often near-duplicates, and the evidence is produced from compressed LLM representations (with additional noise from imperfect assignments). As a result, the *discriminative* information available for distinguishing among candidates is bounded, i.e., $I(Z;O)$ is small. By Fano's inequality, a bounded $I(Z;O)$ implies a non-trivial lower bound on the routing error that increases with $\log n_k$. Therefore, to achieve a low misrouting rate (small p_e), it is necessary to control the candidate set size n_k. This aligns with the practical intuition: when many candidates are highly similar, identifying the source becomes close to guessing unless we reduce the arity of the routing problem.

C.2.2 Connection to Metric Designing

Equation (6) shows that if $I(Z;O)$ is bounded (as expected when many candidates are highly similar), then increasing n_k necessarily increases a lower bound on the misrouting probability.

In our group partition $P = \{C_k\}_{k=1}^{K}$ of N component nodes with $n_k = |C_k|$, the typical within-group candidate set size directly governs the arity of routing within a group. Eq. (2) estimates the expected within-group scanning cost as $E[n_{\text{cluster}}] = \frac{1}{N}\sum_{k=1}^{K} n_k^2$, and defines $\text{SparsityScore}(P) = \frac{N^2}{K\sum_{k=1}^{K} n_k^2}$. Maximising $\text{SparsityScore}(P)$ thus controls the typical within-group candidate size (and prevents extremely large groups), which is precisely the regime where the lower bound in Eq. (6) becomes prohibitive. This provides a theoretical motivation for balancing group sizes: it limits unavoidable routing errors under bounded discriminative evidence in high-similarity dialogue memory.

C.2.3 Fano-style Lower Bound and optimal n_k in hierarchical structure.

In this subsection, we use the Fano-style lower bound to guide us to find the optimal n_k.

Theorem C.1. *Let $Z \in \{1, \dots, n_k\}$ denote the (unknown) correct index and let O be the observable evidence used to infer Z. For any estimator $\hat{Z} = g(O)$ with error probability $p_e = \Pr[\hat{Z} \neq Z]$,*

$$H(Z \mid O) \leq h(p_e) + p_e \log_2(n_k - 1), \tag{7}$$

Table 8: **Effect of the group size cap on LoCoMo (Qwen3-8B).** We vary the upper bound of memory component nodes per group and report BLEU/F1 by question category and the average, together with resulting structure statistics.

Upper bound	Multi-hop		Temporal		Open-domain		Single-hop		Average		# Com.	# Groups	Avg. com. per group
	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1			
14	26.31	37.44	28.67	36.33	13.78	18.41	39.43	48.57	33.19	42.10	2861	383	7.47
12	**27.24**	**38.57**	**29.58**	**37.46**	**15.55**	**20.69**	**40.94**	**50.94**	**34.48**	**43.98**	2879	642	4.48
10	26.52	37.62	28.93	36.78	14.12	18.02	39.31	49.66	33.23	42.80	2932	805	3.64
8	25.01	36.31	27.46	35.24	12.91	17.82	38.49	47.81	32.13	41.21	2862	1111	2.58

where $h(\cdot)$ is the binary entropy function. In particular, if Z is uniform on $\{1, \ldots, k\}$, then

$$p_e \geq 1 - \frac{I(Z;O) + 1}{\log_2 n_k}. \tag{8}$$

Proof. Equation (7) is the standard form of Fano's inequality. Using $I(Z;O) = H(Z) - H(Z \mid O)$ and rearranging yields

$$H(Z) - I(Z;O) \leq h(p_e) + p_e \log_2(n_k - 1).$$

Since $h(p_e) \leq 1$ and $\log_2(n_k - 1) \leq \log_2 n_k$, we have

$$H(Z) - I(Z;O) \leq 1 + p_e \log_2 n_k.$$

If Z is uniform, then $H(Z) = \log_2 n_k$, hence

$$\log_2 n_k - I(Z;O) \leq 1 + p_e \log_2 n_k \quad \Rightarrow \quad p_e \geq 1 - \frac{I(Z;O) + 1}{\log_2 n_k},$$

which is (8). □

Corollary C.2 (Admissible candidate set size under bounded discriminability)**.** *Assume the routing evidence has bounded discriminative information $I(Z;O) \leq B$ (in bits). If we require $p_e \leq \varepsilon$ for some $\varepsilon \in (0,1)$, then any feasible n_k must satisfy*

$$\log_2 n_k \leq \frac{B+1}{1-\varepsilon} \quad \Longrightarrow \quad n_k \leq 2^{\frac{B+1}{1-\varepsilon}}. \tag{9}$$

Proof. Rearrange (8) and substitute $I(Z;O) \leq B$. □

To instantiate Corollary C.2, we need a conservative upper bound B on the mutual information $I(Z;O)$ between the routing evidence and the correct candidate index. Within a group of near-duplicate components, the summary signal O carries only a few bits of discriminative information beyond what the components share; we therefore adopt $B = 2$ bits, an upper-bounding assumption that lets an oracle perfectly distinguish at most $2^B = 4$ candidates without further side information. With a routing accuracy target of $\alpha = 1 - \varepsilon = 0.85$, Corollary C.2 yields

$$n_k \;\leq\; 2^{(B+1)/\alpha} \;=\; 2^{3.529} \;\approx\; 11.5,$$

which we round to $n_k^{\max} = 12$ as the split threshold in our implementation.

This information-theoretic prediction is corroborated empirically. Table 8 sweeps the cap from 8 to 14 on LoCoMo with Qwen3-8B and shows that $n_k^{\max} = 12$ achieves the best BLEU/F1 in *every* question category and the best average ($34.48/43.98$). Caps above 12 (e.g., 14) reduce performance because larger groups exceed the routing-discriminability budget predicted by Fano, while caps below 12 (e.g., 10 or 8) over-fragment the hierarchy and weaken multi-fact coverage.

The actual average branching factor under the cap of 12 is ≈ 4.5 components per group. The cap is a ceiling rather than a target: the SparsityScore + SemScore objective in Eq. (1) produces a distribution of group sizes well concentrated below $n_k^{\max}$, with only the largest groups approaching the Fano-admissible limit. The two mechanisms are complementary: Fano supplies a worst-case ceiling that prevents pathologically large groups, while the objective handles fine-grained sizing within the safe regime.

Table 9: **Expanded results for evidence density and coverage efficiency on LoCoMo (Qwen3-8B).** We report BLEU and F1 for each question category and the average. *Avg. blocks for coverage* is the average number of retrieved blocks required to cover all answer content tokens, and *Cover token cost/query* is the corresponding token cost (lower is better).

Framework	Multi-hop		Temporal		Open-domain		Single-hop		Average		Avg. blocks↓ for coverage	Avg. tokens↓ for coverage
	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1		
Naive RAG	22.46	34.28	17.21	21.32	12.35	17.08	35.68	45.22	27.95	36.48	10.81	1979.26
RAG + Pruning	23.81	33.58	16.53	23.33	12.92	16.22	32.85	41.23	26.55	34.58	13.31	1587.99
xMemory (Ours)	**27.24**	**38.57**	**29.58**	**37.46**	**15.55**	**20.69**	**40.94**	**50.94**	**34.48**	**43.98**	**5.66**	**974.56**

Table 10: **Extended results for retroactive restructuring on LoCoMo (Qwen3-8B).** We report BLEU/F1 by question category and the average, together with the dynamic reassignment ratio under each construction setting.

Setting	Multi-hop		Temporal		Open-domain		Single-hop		Average		Reassign ratio
	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	
w/o merge&split	24.01	34.35	28.24	34.90	10.97	14.04	37.24	44.23	31.30	38.59	0.00%
w/o split	24.46	35.46	28.55	34.77	11.61	15.75	37.69	44.79	31.73	39.18	17.33%
w/o merge	24.81	35.46	29.68	35.99	13.09	18.09	38.45	46.16	32.54	40.33	26.23%
xMemory (Full)	**27.24**	**38.57**	**29.58**	**37.46**	**15.55**	**20.69**	**40.94**	**50.94**	**34.48**	**43.98**	**44.91**%

C.3 Full Performance in Evidence Density under RAG and Pruning Effect

Table 9 provides the detailed performance and coverage statistics for the three settings on LoCoMo with Qwen3-8B. RAG with pruning shows mixed behaviour across categories: while it slightly improves BLEU on multi-hop and open-domain questions, it reduces F1 and consistently underperforms Naive RAG on single-hop and the overall average. This is consistent with Figure 5, where pruning shifts retrieved blocks from 2-hit and multi-hit toward 1-hit, suggesting that compression often discards answer-bearing details. In contrast, xMemory improves both BLEU and F1 in every category, with particularly large gains on temporal and single-hop questions, and achieves the highest overall average. It also covers answer evidence more efficiently, requiring fewer blocks for full coverage (5.66 vs. 10.81 and 13.31) and substantially lower coverage token cost (974.56 vs. 1979.26 and 1587.99), indicating more answer-sufficient retrieval under a tighter budget.

C.4 Full Performance of Retroactive Restructuring in Memory Construction

Extended results. Table 10 provides category-level QA results together with the dynamic reassignment ratio for different construction settings. Table 11 reports the resulting numbers of group and memory component nodes.

Summary. Disabling both operators yields a static hierarchy with 0% reassignment and the lowest overall accuracy. Allowing split or merge enables retroactive reassignment and improves QA, while the full system achieves the highest reassignment ratio and the best accuracy across all categories. Split accounts for a larger share of reassignment, whereas merge is important for consolidating redundant groups and producing a compact high-level organisation, as reflected by the reduced number of groups under the full setting.

C.5 Additional Cost–Performance Statistics

Table 12 reports the token and performance statistics used in Figure 6. Following the LightMem-style reporting setup [7], We report the average construction cost and inference-time token cost per query, and average BLEU/F1 on LoCoMo with Qwen3-8B. Methods marked with ^{F} are flat retrieval baselines, and methods marked with ^{S} are structured memory baselines. The table shows that xMemory introduces moderate construction overhead compared with Nemori, but substantially reduces inference-time token cost while achieving the highest BLEU and F1.

Table 11: **Structure statistics under different construction settings.** We report the resulting numbers of group and memory component nodes.

Setting	#Groups	#Components
w/o merge&split	1114	2876
w/o split	651	2897
w/o merge	1122	2928
Full	642	2879

Table 12: **Token and performance statistics for the cost–performance analysis on LoCoMo with Qwen3-8B.** Construction Token denotes the memory construction cost in thousands of tokens, following the LightMem-style reporting setup. Token/query denotes the average inference-time token cost per query. Methods marked with ^{F} are flat retrieval baselines, and methods marked with ^{S} are structured memory baselines.

Method	**Construction Token (k)**	**Token /query**	**Avg BLEU**	**Avg F1**
Full Memory^{F}	0.00	18535.90	25.83	33.54
Naive RAG^{F}	0.00	8633.28	27.95	36.48
LightMem^{F}	125.95	5545.35	26.04	33.66
Nemori^{S}	381.29	7754.66	31.44	40.88
A-Mem^{S}	1008.55	9103.46	27.84	37.13
MemoryOS^{S}	481.89	7234.66	29.20	33.76
xMemory **(Ours)**	421.91	**4711.29**	**34.48**	**43.98**

C.6 Semantic Answer Evaluation

To complement lexical-overlap metrics, we use GPT-5-mini as an LLM judge to assess the semantic correctness of generated answers against the reference answers. We report the percentage of answers judged correct (Accuracy, %). This evaluation covers all three reader models on LoCoMo and PerLTQA. We additionally include Mem0 [28] as a comparator in this analysis, beyond the six baselines in the main lexical-metric tables.

As shown in Table 13, xMemory achieves the highest judged semantic accuracy in all six dataset–reader combinations, improving upon the strongest comparator by 1.81–4.02 percentage points. This corroborates the improvements observed with BLEU-1, token-level F1, and ROUGE-L, suggesting that the gains are not limited to lexical overlap with the reference answers.

C.7 Sensitivity to Embedding and Decoding Settings

We further examine whether the performance gains of xMemory depend on the choice of embedding model or decoding temperature. Both experiments are conducted on LoCoMo with Llama-3.1-8B-Instruct, using GPT-5-mini to evaluate semantic answer accuracy.

Sensitivity to embedding models. We replace the default `text-embedding-3-small` with BGE-M3 and rerun all frameworks that use embeddings, while keeping the remaining experimental settings unchanged. Full Memory is unaffected by this change since it does not use embedding-based retrieval.

As shown in Table 14, xMemory achieves the highest overall accuracy of 45.91%, outperforming the strongest baseline by 3.90 percentage points. Compared with its default embedding configuration (45.52%), its accuracy remains stable. These results suggest that the performance advantage of xMemory is not specific to the default embedding model.

Table 13: **Semantic answer accuracy (%) judged by GPT-5-mini** on LoCoMo and PerLTQA. Each score is the percentage of answers judged correct against the reference answers. Higher is better, and the best result in each dataset–reader setting is in bold.

Method	LoCoMo			PerLTQA		
	Qwen3-8B	Llama-3.1-8B-Ins	GPT-5 nano	Qwen3-8B	Llama-3.1-8B-Ins	GPT-5 nano
Full Memory	42.40	36.95	48.38	44.61	44.91	47.50
Naive RAG	44.29	39.35	50.78	43.50	45.28	49.70
LightMem	43.44	37.14	50.97	41.18	40.91	46.50
Mem0	47.01	40.26	51.36	45.00	44.23	49.18
A-Mem	45.19	40.91	51.23	42.80	45.40	50.90
MemoryOS	47.40	39.29	52.01	45.60	40.32	45.80
Nemori	47.99	42.79	52.99	46.20	45.00	51.30
xMemory **(Ours)**	**52.01**	**45.52**	**56.04**	**49.70**	**47.50**	**53.11**

Table 14: **Sensitivity to embedding models** on LoCoMo with Llama-3.1-8B-Instruct. All results are GPT-5-mini-judged semantic accuracy (%) using BGE-M3 embeddings. Best results are highlighted in bold.

Method	Multi-hop	Temporal	Open-domain	Single-hop	Average
Full Memory	33.33	22.12	18.75	45.90	36.95
Naive RAG	38.65	22.74	22.92	49.70	40.39
LightMem	35.11	18.07	19.79	45.18	36.10
Mem0	40.43	23.68	25.00	46.37	39.22
Nemori	42.20	**31.15**	26.04	47.92	42.01
A-Mem	37.94	26.17	21.88	50.30	41.23
MemoryOS	36.52	26.17	27.08	48.51	40.32
xMemory **(Ours)**	**48.58**	30.84	**32.29**	**52.32**	**45.91**

Sensitivity to decoding temperature. We further evaluate all frameworks using a decoding temperature of $T = 0.6$, instead of the default deterministic setting ($T = 0$), while keeping the default embedding model and other settings unchanged.

As shown in Table 15, xMemory maintains the highest overall accuracy of 46.10%, exceeding the strongest baseline by 2.27 percentage points. Compared with its default deterministic setting (45.52%), the accuracy increases slightly. This suggests that the overall advantage of xMemory is preserved under the alternative decoding temperature.

D Case Study: Decoupling before Aggregation

To illustrate the memory constructed by xMemory, Figure 7 shows a qualitative example from LoCoMo. The query asks: *"When did Gina lose her job at DoorDash?"*, whose gold answer is *"January 2023."* The original memory stream contains several related events: Gina mentions losing her DoorDash job in January, later refers again to the job loss, and subsequently discusses opening and promoting an online clothing store. The challenge is therefore not that relevant information is missing, but that the decisive temporal evidence is embedded within a cluster of highly similar memories.

The left side of Figure 7 shows how flat similarity-based retrieval behaves over this raw memory stream. Since later memories are semantically close to the query, top-k retrieval selects broadly relevant memories about Gina's job loss and business transition. However, these memories mix the original job-loss event with its later consequences, so the temporal anchor is not clearly isolated and the reader produces a vague answer.

Table 15: **Sensitivity to decoding temperature** on LoCoMo with Llama-3.1-8B-Instruct. All results are GPT-5-mini-judged semantic accuracy (%) at $T = 0.6$. Best results are highlighted in bold.

Method	Multi-hop	Temporal	Open-domain	Single-hop	Average
Full Memory	34.75	25.55	19.79	48.75	39.55
Naive RAG	38.30	24.92	20.83	51.01	41.36
LightMem	35.46	21.81	21.88	47.44	38.31
Mem0	38.65	24.92	22.92	45.78	38.70
Nemori	43.97	**34.89**	28.12	48.99	43.83
A-Mem	35.82	25.55	19.79	48.87	39.81
MemoryOS	37.59	26.79	28.13	48.75	40.84
xMemory **(Ours)**	**47.52**	33.64	**29.17**	**52.32**	**46.10**

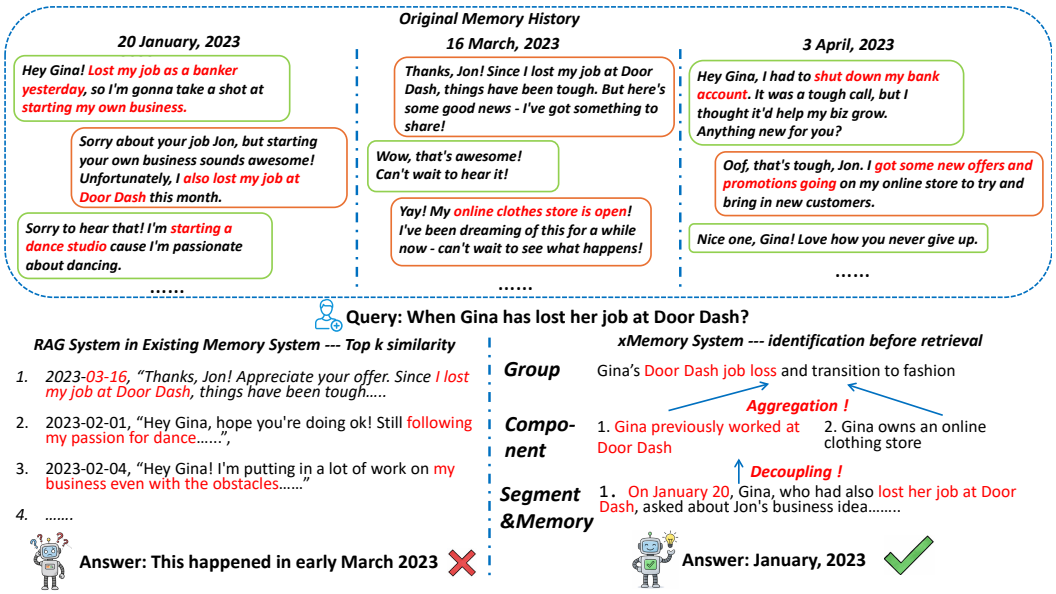

Figure 7: **Case study on LoCoMo.** The query asks *"When did Gina lose her job at DoorDash?"* The original memory stream contains several similar events about Gina's DoorDash job loss and later clothing-store transition. Flat similarity retrieval selects broadly relevant memories from this cluster, but mixes the original job-loss event with later consequences. In contrast, xMemory organises the same history into a hierarchy: a group provides coarse access to the relevant memory region, components separate the DoorDash job-loss fact from related business-transition facts, and the linked source segment preserves the January evidence needed for answer generation.

The right side of Figure 7 illustrates how xMemory represents the same evidence through its hierarchy. The group provides compact access to Gina's employment and business-transition memories. Within this group, separate memory components distinguish related but different facts, such as Gina's prior DoorDash job and her later online clothing store. The selected component then points back to the source segment, where the January interaction provides the original local context and temporal evidence.

This example illustrates why the hierarchy is not merely a compressed summary of the raw history. Groups provide coarse access to related evidence regions, components preserve distinctions among similar events, and source-linked segments retain the original context needed for final answer generation.

E Limitations

This work focuses on retrieval and memory organisation for long-term agent memory, and its empirical evaluation is conducted on LoCoMo and PerLTQA. Although these benchmarks cover multi-session dialogue and personalised long-term memory, they do not exhaust all possible agent-memory settings. We therefore view extending the evaluation to broader agentic settings as an important direction for future work.

xMemory also introduces additional memory-construction cost because it extracts memory components and maintains a revisable hierarchy. Our experiments include an end-to-end cost–performance analysis that accounts for construction and inference tokens, but the practical trade-off may depend on the frequency of memory updates and future queries. In applications with very few queries after memory construction, simpler retrieval methods may be preferable.

Finally, the current implementation relies on LLM-generated memory components and summaries. While the structure is designed to reduce redundancy and preserve links to source segments, errors in component extraction may still affect downstream retrieval. Future work could investigate more robust extraction, verification, and privacy-preserving memory maintenance.

F Broader Impact Statement

This work advances agent-memory retrieval for long-horizon interactions by reducing redundant context while preserving temporally linked evidence. It may improve the reliability and efficiency of applications such as personal assistants, long-term dialogue systems, and multi-session decision-support tools, while reducing inference cost through lower token usage.

Potential risks include privacy leakage, unintended retention of sensitive user information, and harmful reliance on outdated or incorrect memories when deployed with real user data. More effective memory retrieval may also increase the importance of responsible memory management, since retrieved information can influence downstream agent behaviour. Our work does not introduce new data collection and is evaluated on existing benchmarks, but practical deployments should incorporate user consent, data minimisation, access control, retention and deletion policies, and mechanisms for users to inspect, correct, or override stored memories. Overall, we do not anticipate societal impacts beyond those commonly associated with retrieval-augmented LLM systems, but responsible use requires careful privacy and safety safeguards.

G Prompt Design

In this section, we present the specific prompts used for the experiments in Section 4.1.

G.1 Memory Structure Management

This appendix lists the prompts used to construct our hierarchical memory from fine to coarse granularity. We first apply a boundary detector to segment the dialogue stream into coherent segments. For each segment, we generate an episodic memory record with a title, third-person narrative content, and an explicit timestamp. We then distill high-value, persistent memory components from accumulated segments to form reusable long-term knowledge, filtering out transient conversation details. Finally, we summarise clusters of related memory component statements into concise group descriptions that serve as stable high-level indices for structure management and retrieval.

G.2 Answer Generation

We provide dataset-specific answer generation prompts to match the distinct answer formats and evaluation protocols of LoCoMo and PerLTQA. For LoCoMo, the prompt enforces short phrase-style outputs and explicit handling of temporal reasoning by resolving relative time expressions using memory timestamps. For PerLTQA, the prompt requests a single complete sentence and prioritises explicit factual attributes from the memory store, resolving conflicts by recency. Importantly, to ensure fair comparison, we keep the final answer prompt identical across different baselines within each dataset.

Prompt for Boundary Detect

You are a dialogue boundary detection expert. You need to determine if the newly added dialogue should end the current episode and start a new one as a segment.
Current conversation history: $\{conversation\ history\}$
Newly added messages: $\{new\ messages\}$
Please carefully analyze the following aspects to determine if a new segment should begin:
1. **Topic Change** (Highest Priority): - Do the new messages introduce a completely different topic? - Is there a shift from one specific event to another? - Has the conversation moved from one question to an unrelated new question?
2. **Intent Transition**: - Has the purpose of the conversation changed? (e.g., from casual chat to seeking help, from discussing work to discussing personal life) - Has the core question or issue of the current topic been answered or fully discussed?
3. **Temporal Markers**: - Are there temporal transition markers ("earlier", "before", "by the way", "oh right", "also", etc.)? - Is the time gap between messages more than 30 minutes?
4. **Structural Signals**: - Are there explicit topic transition phrases ("changing topics", "speaking of which", "quick question", etc.)? - Are there concluding statements indicating the current topic is finished?
5. **Content Relevance**: - How related is the new message to the previous discussion? (Consider splitting if relevance < 30- Does it involve completely different people, places, or events?
Decision Principles: - **Prioritize topic independence**: Each episodic segment should revolve around one core topic or event - **When in doubt, split**: When uncertain, lean towards starting a new episodic segment - **Maintain reasonable length**: A single episodic segment typically shouldn't exceed 10-15 messages
Note: - If conversation history is empty, this is the first message, return false - When a clear topic change is detected, split even if the conversation flows naturally - Each episodic segment should be a self-contained conversational unit that can be understood independently

Prompt for Episodic Generation

You are an episodic segment memory generation expert. Please convert the following conversation into an episodic segment memory.
Conversation content: $conversation$
Boundary detection reason: $boundary\ reason$
Please analyze the conversation to extract time information and generate a structured episodic memory. Return containing the following three fields: "title": "A concise, descriptive title that accurately summarises the group (10-20 words)", "content": "A detailed description of the conversation in third-person narrative. It must include all important information: who participated in the conversation at what time, what was discussed, what decisions were made, what emotions were expressed, and what plans or outcomes were formed. Write it as a coherent story so that the reader can clearly understand what happened. Ensure that time information is precise to the hour, including year, month, day, and hour.", "timestamp": "YYYY-MM-DDTHH:MM:SS format timestamp representing when this episodic segment occurred (analyse from message timestamps or content)"
Time Analysis Instructions: 1. **Primary Source**: Look for explicit timestamps in the message metadata or content 2. **Secondary Source**: Analyze temporal references in the conversation content ("yesterday", "last week", "this morning", etc.) 3. **Fallback**: If no time information is available, use a reasonable estimate based on context 4. **Format**: Always return timestamp in ISO format: "2024-01-15T14:30:00"
Requirements: 1. The title should be specific and easy to search (including key topics/activities). 2. The content must include all important information from the conversation. 3. Convert the dialogue format into a narrative description. 4. Maintain chronological order and causal relationships. 5. Use third-person unless explicitly first-person. 6. Include specific details that aid keyword search. 7. Notice the time information, and write the time information in the content. 8. When relative times (e.g., last week, next month, etc.) are mentioned in the conversation, you need to convert them to absolute dates (year, month, day). Write the converted time in parentheses after the original time reference. 9. **IMPORTANT**: Analyze the actual time when the conversation happened from the message timestamps or content, not the current time.
Example: If the conversation is about someone planning to go hiking and the messages have timestamps from March 14, 2024 at 3:00 PM: "title": "Weekend Hiking Plan March 16, 2024: Sunrise Trip to Mount Rainier", "content": "On March 14, 2024 at 3:00 PM, the user expressed interest in going hiking on the upcoming weekend (March 16, 2024) and sought advice. They particularly wanted to see the sunrise at Mount Rainier, having heard the scenery is beautiful. When asked about gear, they received suggestions including hiking boots, warm clothing (as it's cold at the summit), a flashlight, water, and high-energy food. The user decided to leave at 4:00 AM on Saturday, March 16, 2024 to catch the sunrise and planned to invite friends for the adventure. They were very excited about the trip, hoping to connect with nature.", "timestamp": "2024-03-14T15:00:00"

Prompt for Memory Component Generation

You are an AI memory system. Extract HIGH-VALUE, PERSISTENT semantic memories as memory components from the following segments.
CRITICAL: Focus on extracting LONG-TERM VALUABLE KNOWLEDGE, not temporary conversation details.
Segments to analyze: segments
HIGH-VALUE Knowledge Criteria
Extract ONLY knowledge that passes these tests: - **Persistence Test**: Will this still be true in 6 months? - **Specificity Test**: Does it contain concrete, searchable information? - **Utility Test**: Can this help predict future user needs? - **Independence Test**: Can be understood without conversation context?
HIGH-VALUE Categories (FOCUS ON THESE):
1. **Identity & Professional** - Names, titles, companies, roles - Education, qualifications, skills
2. **Persistent Preferences** - Favorite books, movies, music, tools - Technology preferences with reasons - Long-term likes and dislikes
3. **Technical Knowledge** - Technologies used (with versions) - Architectures, methodologies - Technical decisions and rationales
4. **Relationships** - Names of family, colleagues, friends - Team structure, reporting lines - Professional networks
5. **Goals & Plans** - Career objectives - Learning goals - Project plans
6. **Patterns & Habits** - Regular activities - Workflows, schedules - Recurring challenges
Examples:
HIGH-VALUE (Extract these): - "Caroline's favorite book is 'Becoming Nicole' by Amy Ellis Nutt" - "The user works at ByteDance as a senior ML engineer" - "The user prefers PyTorch over TensorFlow for debugging" - "The user's team lead is named Sarah" - "The user is learning Rust for systems programming" - "The user has been practicing yoga since March 2021" - "The user joined Amazon in August 2020 as a data scientist" - "The user plans to relocate to Seattle in January 2025"
LOW-VALUE (Skip these): - "The user thanked the assistant" - "The user was confused about X" - "The user appreciated the help" - "The conversation was productive" - Any temporary emotions or reactions
Quality over quantity - extract only knowledge that truly helps understand the user long-term.

Prompt for group Generation

"You are building a stable high-level group summary that captures recurring facts." "Given the following factual statements, write a concise abstract group that names the topic:" f"joined" "Return only the summary."

Prompt for Answer Generation in LOCOMO Dataset

You are an intelligent memory assistant tasked with retrieving accurate information from conversation memories.
CONTEXT: You have access to memories from two speakers in a conversation. These memories contain timestamped information that may be relevant to answering the question.
INSTRUCTIONS: 1. Carefully analyze all provided memories from both speakers 2. Pay special attention to the timestamps to determine the answer 3. If the question asks about a specific event or fact, look for direct evidence in the memories 4. If the memories contain contradictory information, prioritize the most recent memory 5. If there is a question about time references (like "last year", "two months ago", etc.), calculate the actual date based on the memory timestamp. For example, if a memory from 4 May 2022 mentions "went to India last year," then the trip occurred in 2021. 6. Always convert relative time references to specific dates, months, or years. For example, convert "last year" to "2022" or "two months ago" to "March 2023" based on the memory timestamp. Ignore the reference while answering the question. Do not answer temporal questions in timestamp such as "2023-05-08", but answer naturally such as "7 May 2023" or "The week before 6 July 2023"! 7. Focus only on the content of the memories from both speakers. Do not confuse character names mentioned in memories with the actual users who created those memories. 8. The answer should be less than 5-8 words.
Component Memories: components
Segment Memories: segments
Question: question
Answer:

Prompt for Answer Generation in PerLTQA Dataset

You are a memory question-answering assistant. Answer the question using ONLY the information in the provided memories.
Memories may contain profiles, events, and relationship descriptions. Prefer explicit facts (e.g., "Gender: female", "Nationality: China", dates, numbers, names). If multiple memories conflict, choose the most recent one by timestamp.
OUTPUT RULES (VERY IMPORTANT): - Write ONE concise, complete sentence. - Preserve the key value EXACTLY as stated in the memories (names, numbers, dates, gender terms, nationality words). - Do NOT add extra explanation. - Do NOT include quotes, bullet points, or prefixes like "Answer:".
Segment Memories: Segment
Component Memories: component
Question: question
Answer:

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's main scope: identifying the mismatch between standard RAG and agent memory, proposing xMemory as a revisable hierarchical memory framework, and evaluating it on LoCoMo and PerLTQA. These claims are supported by the method description, main results, and analyses in Sections 3–5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper includes a Fano-style theoretical motivation for controlling the candidate set size in hierarchical retrieval. The assumptions, theorem statement, corollary, and proof are provided in Appendix C.2.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Both code and dataset are provided. We also report xMemory implementation details in Appendix B and prompts in Appendix G.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide an anonymised code package to reproduce the main experimental results. The paper uses public benchmarks, and the experimental setup, fixed hyperparameters, model choices, decoding settings, and prompts are described in Sections 4.1, Appendix B, and Appendix G.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We cover all dataset details in Section 4.1. We also report the xMemory hyperparameters in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The experiments utilise the API of Large Language Models and the high performance GPUs. Multiple calls will significantly increase costs. Our group resources are limited. To reduce sampling variance, we use deterministic decoding, and we evaluate consistently across two benchmarks and three LLMs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the main compute resource used for model experiments in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses existing public benchmarks and evaluates retrieval methods for agent memory without collecting new human-subject data or releasing high-risk models or datasets. The submission is anonymized, and we have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We discuss broader impacts in Appendix F, including potential benefits for reliable and efficient long-term memory retrieval, as well as risks related to privacy leakage, sensitive information retention, and reliance on outdated or incorrect memories.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk models, scraped datasets, or sensitive data assets. We release code for a retrieval and memory-organisation framework and evaluate it on existing public benchmarks; broader privacy and deployment considerations are discussed in Appendix F.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the original papers for the datasets, models, and baseline methods used in our experiments, and use them for research evaluation under their respective terms. We discuss in the Appendix B.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We provide an anonymised code package for the proposed method together with setup, data preparation, and experiment-running instructions. No new dataset or trained model is introduced.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve new crowdsourcing experiments or research with human subjects. We evaluate on existing public benchmarks and do not collect new participant data or provide participant instructions or compensation.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve new crowdsourcing experiments or research with human subjects. We use existing public benchmarks for algorithmic evaluation, so IRB approval or equivalent human-subjects review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We used LLMs only for language editing purposes.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.