MEMFOLDER: STRUCTURE-GUIDED MEMORY FOLDING FOR EFFICIENT AGENT MEMORY MANAGEMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term agents accumulate large external memory stores as interactions unfold, increasing the costs of storage, retrieval, and response generation. Efficiency is constrained by two coupled factors: a growing record count increases retrieval overhead, while longer retrieved contexts increase prefill computation and decoding-time attention costs. Existing methods prune or consolidate records, but pruning can discard useful evidence, whereas consolidation may produce fewer yet longer retrieval units. We introduce **MemFolder**, a structure-guided memory folding framework that sequentially reduces record count and context length. The framework comprises two core components that share memory groups, using cross-record structure to guide both evidence consolidation and compact representation. **RecordFolder** grounds entity mentions, progressively selects representative yet complementary anchors, and combines semantic and entity affinity with temporal chaining to consolidate compatible records into fewer retrieval units. **ContextFolder** encodes the original records within these groups into compact text and uses verification feedback to revise encodings when key facts cannot be recovered. Experiments on the three text-based memory benchmarks LoCoMo, PersonaMem, and LongMemEval and the two multimodal benchmarks M3-Bench-web and M3-Bench-robot demonstrate the strong performance of our methods. Notably, on Mem0, RecordFolder retains **99.0%** of the original average performance across the three text-based benchmarks with 50% fewer records, exceeding the strongest baseline by **1.9** percentage points in performance retention. Combined with ContextFolder, existing record compression methods further shorten their output records while retaining most of their performance.

1 INTRODUCTION

Advances in reasoning and tool use enable large language models (DeepSeek-AI et al., 2026; Kimi Team et al., 2026a;b) to support long-term agents operating across multiple sessions (Wu et al., 2025; He et al., 2026). These agents rely on accumulated conversations, observations, and task states to inform later decisions. External memory systems (Chhikara et al., 2025; Xu et al., 2025; Liu et al., 2026a) retain this experience as natural-language records indexed for retrieval. However, the stores grow as interactions continue. Across all ten LoCoMo samples (Maharana et al., 2024), our uncompressed stores contain **6,817 records** totaling **61,273 tokens**, making memory management an increasingly important part of practical long-term agent deployment.

This growth creates **two coupled efficiency bottlenecks**, as shown in Figure 1(1). First, a large **record count** increases storage and index size and enlarges the retrieval search space. Second, the **context length** of retrieved records determines how much memory text the agent processes for each query. Longer contexts increase prefill computation and decoding-time attention costs. Even individually short records can impose substantial overhead when many are retrieved together, so efficient memory management must address both quantities.

Existing methods typically address one axis, as summarized in Figure 1(2). Prompt compression (*e.g.*, LLMLingua (Jiang et al., 2023; Pan et al., 2024) and BabelTele (Zhu et al., 2026)) shortens the input to an individual call while leaving N unchanged, whereas StreamMeCo (Wang et al., 2026a) and MemForest (Wang et al., 2026b) compress stored records through pruning or consolidation. Pruning can discard useful evidence, whereas consolidation preserves related content but makes

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

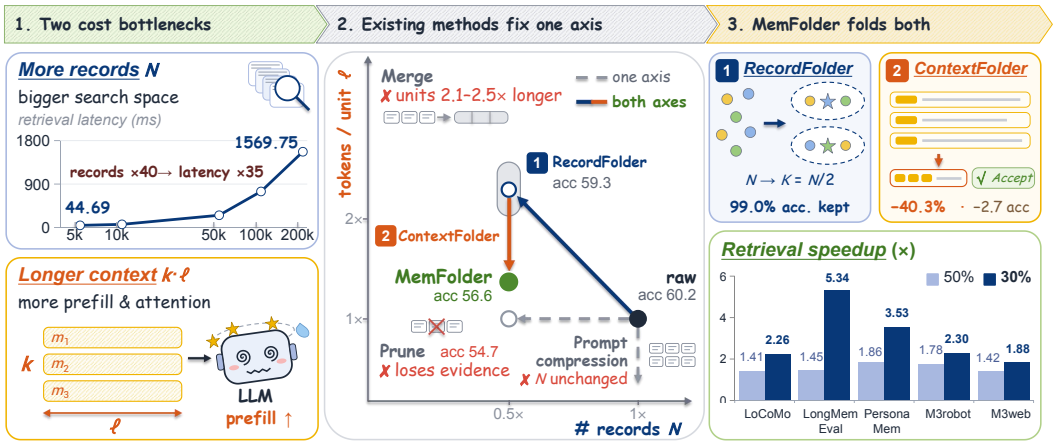


Figure 1: **MemFolder for efficient agent memory management. (1)** More records increase retrieval costs, while longer contexts delay first-token generation. **(2)** Existing methods address one axis, whereas MemFolder combines **RecordFolder** and **ContextFolder** to reduce both. **(3)** Fewer records and shorter contexts preserve accuracy while improving retrieval speed and time to first token.

each surviving retrieval unit longer. Our results show that keeping 50% of the records leaves units averaging $2.12\times$ and $2.48\times$ the mean length of an original record for MemForest and ToMe (Bolya et al., 2023) on LoCoMo (Maharana et al., 2024); at 30% retention, these ratios increase to $3.01\times$ and $3.87\times$. Even the no-merge concatenation control produces units averaging $2.10\times$ the original length at 50% retention. With a fixed top-k retrieval budget, **reducing the number of units therefore does not guarantee a smaller context**. The challenge is to preserve evidence needed for future interactions while reducing both the number of retrieval units and the text carried by each unit.

This observation motivates two distinct decisions. At the **record level**, compression groups related records into shared retrieval units while preserving complementary facts and state changes. At the **context level**, it compresses the records within each group by merging shared descriptions while retaining dates and other record-specific details. The shared groups connect these decisions, but reducing their number does not shorten the text they carry.

We address these two efficiency bottlenecks with **MemFolder**, a structure-guided memory folding framework with two complementary folding components. As shown in Figure 1(3), **RecordFolder** grounds entity mentions, progressively selects representative yet complementary anchors, and combines semantic and entity affinity with temporal chaining to consolidate N records into K retrieval units. Given these groups and their source records, **ContextFolder** generates compact encodings under a target token ratio while keeping one output unit per group; verification feedback guides revisions when key facts are not recoverable. Given that it is independent of the grouping algorithm, ContextFolder can also shorten the contexts from record compression methods.

Extensive experiments on three text benchmarks and two multimodal benchmarks show that RecordFolder is able to retain **99.0%** of average Mem0 performance with 50% fewer records, exceeding second-best method MemForest by **1.9%** of the original performance. Combined with RecordFolder, ContextFolder further cuts tokens per record by 40%, 38%, and 31% on LoCoMo, LongMemEval, and M3-Bench-robot, respectively, with limited accuracy loss. As a plug-and-play component, it also shortens outputs from other record-compression methods, yielding more favorable accuracy–efficiency trade-offs than BabelTele and LLMLingua-2. The **main contributions** are as follows:

1. We introduce MemFolder, a structure-guided memory folding framework that jointly reduces the number of retrieval units and their context length for long-term agents.
2. We develop RecordFolder to consolidate records through progressive anchor selection and entity-aware temporal chaining. ContextFolder then compacts each group's content through budgeted encoding and factual recovery checks that guide iterative revisions.
3. Experiments on three text benchmarks and two multimodal settings demonstrate strong performance under record compression, while context compression analyses show further reductions in unit length across multiple existing memory consolidation methods.

2 Related Work

LLM Agent Memory Systems. Memory systems preserve information from past interactions for use beyond the current context window. Token-level external memory retains explicit records that can be inspected, updated, and retrieved without modifying model parameters. Generative Agents (Park et al., 2023) stores observations and reflections in a memory stream, while MemGPT (Packer et al., 2023) manages information across memory tiers. Mem0 (Chhikara et al., 2025) extracts and consolidates salient facts from conversations, and A-MEM (Xu et al., 2025) organizes structured notes through dynamically generated links. Other approaches maintain experience in latent representations or model parameters. MemoryLLM (Wang et al., 2024) uses a fixed-size, self-updatable latent memory pool, whereas TMEM (Ren et al., 2026) incorporates distilled experience through online updates to fast LoRA weights. These forms can coexist: MemOS (Li et al., 2025) unifies the management of plaintext, activation-based, and parameter-level memories. Our work focuses on accumulated external records, whose count and textual content determine the size of the store and the evidence available for retrieval. Rather than introducing another memory architecture, MemFolder compresses these records within existing systems while preserving text for retrieval and generation.

Long-Term Memory Compression. Memory efficiency can be improved during construction, maintenance, or inference. MemoryBank (Zhong et al., 2024) updates retention through forgetting and reinforcement, while SimpleMem (Liu et al., 2026a) constructs compact memory units and recursively consolidates related content. At inference time, LLMLingua (Jiang et al., 2023) shortens prompts, whereas KV cache methods (Zhang et al., 2023; Li et al., 2024; Liu et al., 2026b) reduce the states retained during generation. These inference-side interventions do not by themselves reduce the number of records in a persistent external store. We study post-construction compression before retrieval, so compressed records can be reused across queries.

Post-construction methods reduce existing stores through pruning or consolidation. StreamMeCo (Wang et al., 2026a) prunes memory graphs using representative sampling and scores based on entity importance and semantic diversity, without rewriting the retained records. MemRefine (Kim et al., 2026) uses embedding similarity to propose candidate pairs, then an LLM judge determines whether their factual content warrants deletion, merging, or preservation. Its refinement loop targets a storage budget measured over serialized entries. MemForest (Wang et al., 2026b) combines global semantic similarity with local temporal continuity to partition memories into EventTrees, then progressively merges records along selected tree edges. It also introduces anchor-guided propagation retrieval to supplement evidence with temporally related memories.

MemFolder targets both record count and per-unit context length. Compared with EventTree-based merging (Wang et al., 2026b), RecordFolder groups records around progressively selected anchors and refines assignments through entity-aware temporal chains. ContextFolder reuses these groups to generate compact encodings under a soft token budget. It checks whether source-derived facts can be recovered from the encoding without exposing the original records to the reader, and uses the feedback for revision. This post-encoding check differs from deciding whether a candidate pair should be merged: it evaluates factual recovery from the resulting compact representation.

3 Methodology

3.1 Preliminary: Agent Memory Systems and Compression

Agent Memory Systems. An agent memory system converts an interaction history $\mathcal{H}_t$ into a persistent external store $\mathcal{M}_t = \{m_i\}_{i=1}^{N_t}$, where N_t is the number of stored records. When a new interaction x_t arrives, an extractor $\mathcal{E}$ produces candidate records $\Omega_t = \mathcal{E}(x_t, \mathcal{H}_{<t})$, and an updater $\mathcal{U}$ incorporates them as $\mathcal{M}_t = \mathcal{U}(\mathcal{M}_{t-1}, \Omega_t)$. Given a query q, retrieval returns $\mathcal{R}_k(q, \mathcal{M}_t) = \mathrm{TopK}_{m \in \mathcal{M}_t} s(q, m)$, where $s(q, m)$ measures relevance. The agent conditions its response on q and the retrieved records. As interactions accumulate, the store requires more persistent storage and exposes more candidates to each retrieval operation as the store grows.

Memory Compression. Accumulated agent memories contain redundancy at two levels: redundant **records** across the store and repeated **context** within each record. For a fixed snapshot $\mathcal{M} = \mathcal{M}_t$, let $N = N(\mathcal{M}) = |\mathcal{M}|$ denote its record count. Record compression reduces this count by retaining representative records or consolidating related ones. Given a record keep ratio $\kappa \in (0, 1]$, it targets

$K = \max\{1, \mathrm{round}(\kappa N)\}$ retrieval units. Context compression reduces the total token length $L(\mathcal{M}) = \sum_{m \in \mathcal{M}} \ell(m)$, where $\ell(m)$ denotes the token length of record m, by expressing shared facts once while retaining group-specific details within each retrieval unit.

3.2 MemFolder Overview

Figure 2 presents the pipelines of our MemFolder, which addresses both memory costs by sequentially folding the record store along two axes. **RecordFolder** grounds records, selects representative and complementary anchors, refines group assignments using entity, semantic, and temporal relations, and consolidates each group into one retrieval unit. **ContextFolder** reuses these groups to generate compact grammar-based encodings under group-specific budgets: a planner identifies facts to preserve, a compressor rewrites each group, and a verifier checks recovery from the encoding alone. Thus, retrieval accesses fewer units with less context in each.

3.3 RecordFolder: Record Compression via Progressive Consolidation

Records that refer to the same entities or events can remain in separate retrieval units, splitting related evidence across results. **RecordFolder** maps N records to K units by grounding entities, selecting complementary anchors, and refining assignments with entity-aware affinity and temporal links. Hungarian matching restricts these links to non-branching chains. An LLM then consolidates each multi-record group chronologically, while singleton records remain unchanged.

Entity Grounding. The same entity may appear under different names or pronouns, so surface overlap alone can miss connections between records. We use `gpt-4o-mini` to extract entities, including people, works, activities, and topics, and assign short canonical labels. Within each session, we process records together to resolve references in context, then process sessions chronologically while carrying earlier labels forward in a registry. The prompt reuses labels for known referents and adds qualifiers to distinguish namesakes. We merge residual label variants whose embeddings have cosine similarity ≥ 0.85, retaining one representative per connected component. After removing conversation speakers and synthetic event identifiers, the remaining labels form the entity set E_i of record m_i for affinity computation and later temporal refinement.

Progressive Anchor Selection. Anchors define the K groups that become retrieval units. At each step, we score every remaining record against the anchors already selected. Let x_i be the normalized embedding of record m_i and $D_{ij} = \|x_i - x_j\|_2^2$. We select the first anchor with a density-peak criterion (Rodriguez & Laio, 2014). We compute local density $\rho_i = \exp(-\frac{1}{k}\sum_{j \in \mathrm{kNN}(i)} D_{ij})$ from the $k = \min(5, N-1)$ nearest other records. Separation is $\delta_i = \min_{j:\rho_j > \rho_i} D_{ij}$, the distance to the nearest record with higher density. We choose $a_1 = \arg\max_i \rho_i \delta_i$, favoring a locally dense record separated from denser alternatives. Subsequent selections replace local density with novelty relative to the set A_r of r selected anchors:

$$\rho_i^{(r)} = 1 - \exp\Big(-\frac{1}{k'} \sum_{a \in \mathrm{kNN}_{A_r}(i)} D_{ia}\Big), \qquad \delta_i^{(r)} = \min_{\substack{j \notin A_r \\ \rho_j^{(r)} > \rho_i^{(r)}}} D_{ij}, \tag{1}$$

where $k' = \min(5, |A_r|)$ and $\mathrm{kNN}_{A_r}(i)$ contains the k' nearest selected anchors. When no unselected record has a higher ρ value, we set $\delta_i^{(r)}$ to the maximum distance from i to an unselected record. We select $a_{r+1} = \arg\max_{i \notin A_r} \rho_i^{(r)} \delta_i^{(r)}$ until K anchors are chosen. The novelty term favors candidates farther from existing anchors, encouraging complementary coverage. Nearest-anchor assignment gives the initial partition π_0, where $a_0(i)$ denotes the anchor assigned to m_i for temporal refinement.

Entity-Aware Affinity. The initial partition uses embeddings; entity overlap adds a complementary signal when linking records over time. To downweight common entities, we assign $w(e) = \log(N/\mathrm{df}(e))$, where $\mathrm{df}(e)$ counts the records containing entity e. For an entity set S, let $\mu(S) = \sum_{e \in S} w(e)$ denote its total weight. The shared entity mass is $\mu_{ij} = \mu(E_i \cap E_j)$, and the entity similarity is $e_{ij} = \mu_{ij}/\mu(E_i \cup E_j)$, set to zero when the denominator is zero. We combine entity similarity with cosine similarity $c_{ij} = \cos(x_i, x_j)$ as $\mathrm{affinity}(i,j) = \alpha_{ij} e_{ij} + (1 - \alpha_{ij}) c_{ij}$. The mixing weight is $\alpha_{ij} = \mu_{ij}\sigma_c^2/(\mu_{ij}\sigma_c^2 + \sigma_e^2)$. The signal variances σ_e^2 and σ_c^2 are computed over all earlier–later pairs within a 50-record window before filtering, each floored at 10^{-6}. The resulting affinity gives greater weight to rare shared entities and reduces to cosine similarity when $\mu_{ij} = 0$.

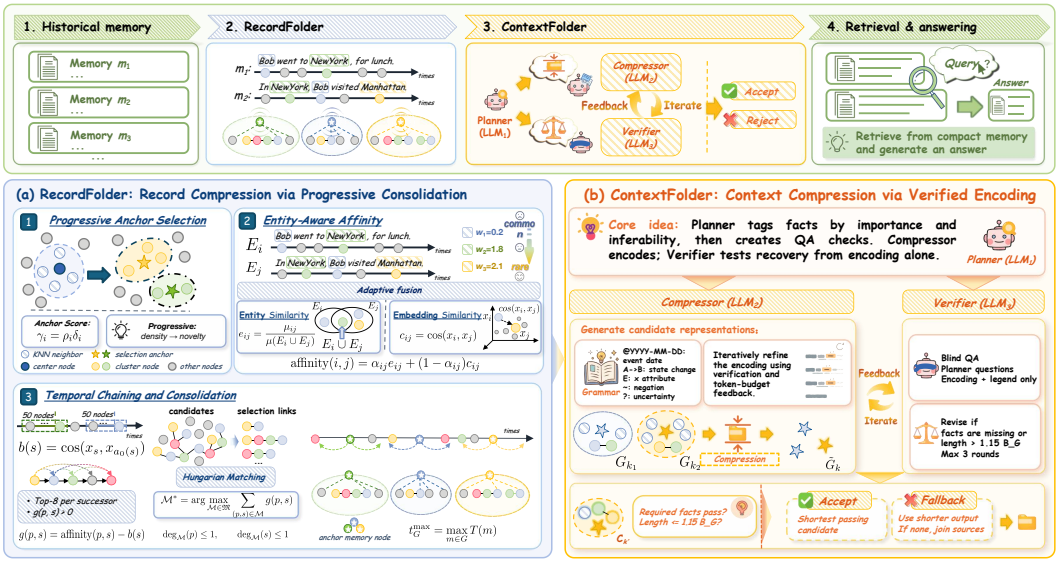


Figure 2: **Overview of MemFolder. (a) RecordFolder** maps N records to K retrieval units through entity grounding, complementary anchor selection, entity-aware affinity, and Hungarian matching of temporal links. **(b) ContextFolder** compresses each group's source records to a target token-retention ratio using a planner that prioritizes facts and prepares checks, a compressor that generates grammar-based candidates, and a blind verifier that tests recovery over up to three rounds. ContextFolder keeps the shortest candidate passing required checks; otherwise, weighted recovery guides selection, with a shorter candidate or source texts used when compression would expand the memory.

Temporal Chaining and Consolidation. We use this affinity to refine the initial assignments by linking records that may continue the same event. For each non-anchor record m_s, we inspect up to 50 predecessors m_p and retain at most 8 links with positive gain $g(p,s) = \text{affinity}(p,s) - \cos(x_s, x_{a_0(s)})$. Thus, a predecessor is retained only when its affinity with m_s exceeds the cosine similarity between m_s and its initial anchor. Hungarian matching maximizes total gain between predecessor and successor copies, allowing each record at most one incoming and one outgoing edge. Because every link follows temporal order, the matched links form non-branching chains. Anchors have no incoming links, so a chain containing an anchor begins with it and remains in its group. We assign each anchor-free chain C to $a^\star(C) = \arg\max_{a \in A_K} \sum_{i \in C} \cos(x_i, x_a)$ and move its members together. Unmatched records retain their initial assignments in π_0, preserving the K groups. An LLM consolidates each multi-record group in one call, reading members chronologically and resolving relative dates from their timestamps. Singleton records remain unchanged, and we embed all resulting units for downstream retrieval and later query answering.

3.4 ContextFolder: Context Compression via Verified Encoding

Consolidating records can reduce the number of retrieval units while lengthening each one; at a fixed retrieval top-k, this need not reduce the context passed to the agent. **ContextFolder** compresses the original records within each assigned group under a target fraction of source tokens, without changing group membership. A planner labels facts by importance and inferability and prepares QA checks. A compressor generates candidate encodings, and a blind verifier tests whether a reader can recover the facts from the encoding alone. Feedback on missing facts and token usage guides revision while preserving each group's assignment and target ratio throughout compression.

Planner: Fact Extraction and Triage. For each group G, an LLM extracts atomic facts from its member records and writes a verification question with an expected answer for each fact. These records are the planner's only source; downstream test queries and answers do not enter the process. The planner labels facts as high, medium, or low importance, prioritizing key events, entity bindings, exact dates and numbers, and negations. It also marks facts inferable from other facts or common knowledge. We assign importance weights of 3, 2, and 1, respectively, and halve the weight of inferable facts when computing weighted recovery. Let $\tau \in (0,1)$ denote the target fraction of

source tokens to retain. Non-inferable high-importance facts are always required; non-inferable medium-importance facts are also required when $\tau \geq 0.65$ for later verification.

Compressor: Ratio-Controlled Grammar Encoding. For a group G, the compressor uses budget $B_G = \max\{3, \text{round}(\tau S_G)\}$, where $S_G = \sum_{m \in G} \ell(m)$ is its total source length. The same τ therefore sets a proportional budget for each group, not a fixed token count. The encoding uses `@YYYY-MM-DD` for event dates, `A -> B` for state changes, and `E: x` for attributes. The symbols `~` and `?` indicate negation and uncertainty. The prompt prioritizes non-inferable, high-importance facts, omitting inferable details first and lower-priority content as needed. For retained facts, it requests exact entity names, numbers, and dates. Both the verifier and downstream agent use the same grammar legend to decode these compact symbols consistently.

Verifier: Blind Recovery Check. Separate answering and judging calls evaluate each candidate. A blind reader receives the encoding, grammar legend, and verification questions, but not the source records or expected answers. It returns `UNKNOWN` when the encoding lacks sufficient evidence. A judge compares the responses with the reference facts, allowing semantic matches while requiring exact names, dates, and numbers. A missing required fact or an encoding longer than $1.15B_G$ triggers feedback for at most three revision rounds. Among candidates that recover all required facts, we select the shortest. If none passes, we select the candidate with the highest weighted recovery and break ties by length. Weighted recovery is the fraction of total fact weight judged recoverable. If the selected encoding is no shorter than S_G, we use the shortest candidate below S_G, even if it misses required facts. If no candidate is shorter, we retain the source texts as one unit.

4 Experiments

4.1 Experimental Setting

Benchmarks. We evaluate MemFolder on four benchmarks: LoCoMo (Maharana et al., 2024), PersonaMem (Jiang et al., 2025), LongMemEval (Wu et al., 2025), and M3-Bench (Long et al., 2026), covering its web and robot subsets. For each benchmark and memory framework, all methods start from the same store constructed from the original interaction histories.

Implementations. All experiments are conducted on two NVIDIA A100 (80GB) GPUs. We use `gpt-4o-mini` for memory node merging. In addition, all other settings, including those of the compared baselines and the underlying memory frameworks, follow the original papers. All experimental results are averaged over three runs. More details are provided in Appendix A.

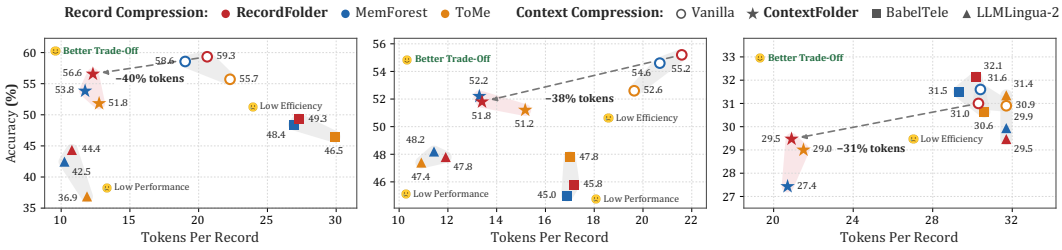
Baselines. For record compression, we compare with pruning baselines (Random Pruning, KMeans, and DART (Wen et al., 2025)) and merging baselines (Random Merging, ToMe (Bolya et al., 2023), and MemForest (Wang et al., 2026b)). For context compression, we compare ContextFolder with LLMLingua-2 (Pan et al., 2024) and BabelTele (Zhu et al., 2026).

4.2 Main Results

Main Comparisons. Table 1 compares RecordFolder with record-compression baselines on Mem0 at 30%, 50%, and 70% reduction. Figure 3(a) and (b) evaluate ContextFolder against context methods after three record compressors at 50% reduction. **(i) RecordFolder leads at every ratio.** It achieves the highest average on all three benchmarks. At 50% reduction, it retains **99.0%** of uncompressed performance, 1.9 points above MemForest; at 70%, it retains **96.6%**, widening the lead to **3.3 points**. **(ii) Consolidation benefits LongMemEval.** At 50% reduction, RecordFolder scores 55.2 on LongMemEval, versus 54.6 for MemForest and 52.6 for ToMe. At 70%, its single-session-preference score rises from 23.3 pre-compression to 53.3, while the average falls from 61.1 to 59.0. We suppose related evidence helps retrieve preference facts together, though this gain does not offset the overall drop. **(iii) ContextFolder improves cross-method trade-offs.** On LoCoMo, BabelTele increases tokens per record after all compressors: after RecordFolder, tokens rise from 20.6 to 27.3 while accuracy drops from 59.3% to 49.3%. We suppose multilingual and symbolic rewriting adds tokenization overhead on compact records, offsetting the text removed. On LongMemEval, BabelTele shortens RecordFolder's output, but ContextFolder achieves higher accuracy with fewer tokens: 51.8% at 13.4 tokens, versus 45.8% at 17.2. It also shortens MemForest and ToMe outputs on both benchmarks. After RecordFolder, it retains 12.2 and 4.0 more accuracy points than LLMLingua-2 for

Table 1: **Record compression on Mem0 cross compression ratios. All**: overall accuracy; **Avg.**: mean accuracy across three benchmarks; **%**: Avg. retained relative to uncompressed Mem0. Best and second-best compressed results per column and ratio are **bold** and underlined, respectively.

Method	LoCoMo					LongMemEval							Persona-Mem	Avg.	%
	SH	MH	TR	OD	All	SU	SA	SP	MS	KU	TR	All			
Mem0	66.0	56.0	53.9	42.7	60.2	90.0	21.4	23.3	54.9	79.5	44.4	55.2	67.9	61.1	100.0
Record-count reduction: 30%															
+ Random Pruning	55.3	49.7	43.3	39.6	50.8	82.9	17.9	26.7	51.9	73.1	39.1	50.8	64.5	55.4	90.7
+ KMeans	61.1	54.3	52.7	40.6	56.8	84.3	17.9	20.0	51.9	75.6	41.4	52.0	65.7	58.2	95.3
+ DART	64.1	53.9	48.0	40.6	57.4	82.9	19.0	20.0	48.9	74.4	34.6	48.8	66.6	57.6	94.3
+ Random Merging	60.9	52.1	51.1	43.8	56.2	**91.4**	**21.4**	20.2	46.6	74.4	41.4	50.8	65.2	57.4	93.9
+ ToMe	63.1	51.1	50.8	44.8	57.2	85.7	**21.4**	23.3	48.1	71.8	42.9	51.2	66.4	58.3	95.4
+ MemForest	63.7	54.6	52.0	**45.8**	58.5	90.0	19.6	20.0	49.8	**76.9**	**49.6**	54.2	66.6	59.7	97.7
+ MemFolder (Ours)	**67.5**	**56.0**	**56.1**	41.7	**61.7**	88.6	**21.4**	**36.7**	**52.6**	75.6	46.6	**55.2**	**67.7**	**61.5**	**100.7**
Record-count reduction: 50%															
+ Random Pruning	44.4	45.4	37.1	**44.8**	43.1	75.7	17.9	26.7	32.3	65.4	39.8	43.6	64.3	50.3	82.3
+ KMeans	51.6	43.6	39.6	37.5	46.8	77.1	19.6	23.3	39.1	74.4	39.8	47.0	65.2	53.0	86.7
+ DART	57.4	49.6	41.7	43.8	51.9	67.1	12.5	23.3	42.9	60.3	31.6	41.4	64.3	52.5	85.9
+ Random Merging	56.0	48.2	46.4	37.5	51.4	85.7	**23.2**	20.0	42.9	70.5	39.1	48.6	64.5	54.8	89.7
+ ToMe	60.2	50.4	47.4	43.8	54.7	85.7	21.4	23.3	48.1	71.8	41.4	50.8	65.9	57.1	93.5
+ MemForest	62.1	**56.7**	**54.8**	43.8	58.4	**90.0**	19.6	20.0	48.1	74.4	42.9	51.8	**67.7**	59.3	97.1
+ MemFolder (Ours)	**65.5**	55.7	50.5	39.6	**59.3**	88.6	21.4	**30.0**	**53.4**	**76.9**	**45.9**	**55.0**	67.2	**60.5**	**99.0**
Record-count reduction: 70%															
+ Random Pruning	35.0	36.5	25.9	31.3	33.1	47.1	7.1	30.0	16.5	42.3	31.6	28.6	63.7	41.8	68.4
+ KMeans	37.9	42.6	27.4	39.6	36.7	61.4	8.9	30.0	21.8	59.0	31.6	34.8	63.8	45.1	73.8
+ DART	46.4	44.3	36.1	41.7	43.6	44.3	10.7	16.7	21.0	43.6	27.8	30.6	61.8	45.3	74.1
+ Random Merging	54.0	47.2	47.0	43.8	50.7	80.0	21.4	30.0	41.4	67.9	34.6	46.2	64.0	53.6	87.7
+ ToMe	56.6	51.8	47.7	38.5	52.7	84.3	21.4	20.0	45.1	70.5	42.1	49.6	**65.7**	56.0	91.7
+ MemForest	58.1	56.4	45.2	**45.8**	54.4	84.3	**25.0**	23.3	**52.6**	74.4	39.8	52.2	64.4	57.0	93.3
+ MemFolder (Ours)	**61.2**	**57.0**	**50.7**	43.4	**57.1**	**85.7**	23.2	**53.3**	50.4	**75.6**	**45.1**	**55.0**	64.9	**59.0**	**96.6**



(a) Performance on LoCoMo (b) Performance on LongMemEval (c) Performance on M3-Bench-robot

Figure 3: **Context compression results.** Accuracy–tokens trade-offs at 50% record-count reduction. Colors and markers denote record and context compression methods. Upper left is better.

about 1.5 extra tokens per record on LoCoMo and LongMemEval. Together, RecordFolder halves the record count, and ContextFolder further reduces tokens per record by **40.3%** and **37.9%**.

Results on Additional Memory Frameworks. Tables 2 and 3 extend RecordFolder to MemOS and M3-Agent. On MemOS, it retains 91.0% and 84.8% of uncompressed performance with 50% and 70% fewer records. On M3-Agent, the average rises from 39.1 without compression to 39.6 and 40.1, reaching 102.6% retention at 70% reduction. Figure 3(c) tests ContextFolder on M3-Bench-robot at 50% record reduction. Across three compressors, BabelTele and LLMLingua-2 barely change the roughly 30-token record-only outputs, while ContextFolder uses 20.7–21.5 tokens per record. After RecordFolder, ContextFolder matches LLMLingua-2 at 29.47% accuracy with 34.1% fewer tokens; BabelTele achieves higher accuracy but fails to meaningfully compress the context. We suppose that short, information-dense event records leave little expendable text for token deletion or symbolic rewriting, whereas group-guided encoding can remove repeated context while checking key facts against source records. The two stages therefore shrink the number and textual footprint of text records within M3-Agent's multimodal long-term memory graph.

Table 2: **Record compression on MemOS across compression ratios.**

Method	LoCoMo					LongMemEval							Persona-Mem	Avg.	%
	SH	MH	TR	OD	All	SU	SA	SP	MS	KU	TR	All			
MemOS	84.5	70.2	78.8	56.2	79.0	91.4	100.0	66.7	55.6	78.2	71.4	74.0	60.3	71.1	100.0
Record compression ratio ↓ 50%															
+ Random Pruning	70.5	65.2	57.3	46.9	65.3	67.1	64.3	63.3	28.6	52.6	43.6	47.8	55.9	56.3	79.2
+ KMeans	75.7	59.2	57.6	57.3	67.8	72.9	69.6	66.7	20.3	51.3	45.9	47.6	54.7	56.7	79.7
+ DART	76.5	68.1	58.9	**58.3**	70.1	62.9	**82.1**	50.0	20.3	35.9	33.1	40.8	53.7	54.9	77.2
+ Random Merging	79.4	69.9	66.0	54.2	73.3	80.0	62.5	**83.3**	**46.6**	60.3	54.9	59.6	56.2	63.0	88.7
+ ToMe	76.7	68.8	63.9	52.1	71.0	78.6	67.9	76.7	40.6	70.5	**63.2**	61.8	**57.9**	63.6	89.4
+ MemForest	**82.6**	68.8	**70.1**	**58.3**	**76.0**	81.4	76.8	66.7	40.6	65.4	56.4	60.0	54.8	63.6	89.5
+ **MemFolder (Ours)**	81.2	**70.6**	69.5	56.2	75.3	**85.7**	78.6	**83.3**	45.1	**75.6**	54.9	**64.2**	54.7	**64.7**	**91.0**
Record compression ratio ↓ 70%															
+ Random Pruning	58.5	59.9	43.0	49.0	54.9	32.9	48.2	73.3	13.5	29.5	28.6	30.2	52.8	46.0	64.7
+ KMeans	62.5	57.8	43.9	55.2	57.3	40.0	53.6	50.0	15.8	30.8	30.1	31.6	52.6	47.2	66.4
+ DART	65.6	59.9	46.1	54.2	59.8	30.0	60.7	50.0	15.0	25.6	18.0	26.8	48.9	45.2	63.5
+ Random Merging	73.0	64.2	61.7	57.3	68.1	**75.7**	39.3	83.3	**39.1**	60.3	**52.6**	53.8	54.0	58.6	82.5
+ ToMe	71.7	68.8	60.7	47.9	67.4	67.1	**67.9**	70.0	32.3	62.8	51.9	53.4	55.4	58.7	82.6
+ MemForest	77.4	64.9	**62.6**	56.2	70.7	62.9	66.1	73.3	30.8	50.0	50.4	50.0	**55.5**	58.7	82.6
+ **MemFolder (Ours)**	**77.5**	**69.1**	60.1	**60.4**	**71.3**	74.3	62.5	**93.3**	33.8	**65.4**	45.9	**54.4**	55.2	**60.3**	**84.8**

Table 3: **Record compression on M3-Bench across compression ratios.**

Method	M3-Bench-robot						M3-Bench-web						Avg.	%
	ME	MH	CM	PU	GK	All	ME	MH	CM	PU	GK	All		
M3-Agent	30.9	29.4	29.6	41.1	22.6	30.3	44.9	25.6	44.8	58.6	53.7	47.9	39.1	100.0
Record compression ratio ↓ 50%														
+ Random Pruning	28.4	25.9	27.3	38.7	19.6	28.1	35.6	23.0	34.9	52.3	44.2	41.2	34.7	88.7
+ KMeans	29.1	30.6	28.4	41.4	20.8	29.2	41.1	21.7	42.5	55.0	47.4	43.9	36.6	93.6
+ DART	28.9	25.9	29.4	39.1	**22.0**	29.1	39.9	23.6	39.4	52.9	47.2	43.1	36.1	92.3
+ StreamMeCo	32.3	28.2	29.8	41.4	21.1	30.6	39.7	25.2	38.9	58.4	47.3	44.7	37.7	96.2
+ Random Merging	**33.0**	32.9	**31.9**	44.3	19.6	30.8	41.8	21.0	40.1	58.0	46.3	44.6	37.7	96.4
+ ToMe	30.5	32.9	29.6	42.3	19.9	30.9	42.7	24.7	41.7	54.3	**53.1**	46.1	38.5	98.5
+ MemForest	31.6	31.8	31.1	45.1	21.4	**31.6**	42.6	23.9	43.4	57.1	51.5	46.5	39.0	99.7
+ **MemFolder (Ours)**	31.7	**34.1**	30.0	**45.3**	19.9	31.0	**44.6**	**27.8**	**44.6**	**59.6**	52.4	**48.2**	**39.6**	**101.3**
Record compression ratio ↓ 70%														
+ Random Pruning	27.3	25.9	26.7	39.2	**20.8**	27.7	35.0	19.9	34.9	48.7	39.5	37.3	32.5	83.1
+ KMeans	28.7	28.2	27.3	42.7	19.3	28.9	35.5	20.1	34.9	39.7	42.5	39.1	34.0	87.0
+ DART	27.8	25.9	25.4	40.0	16.2	27.2	35.4	19.5	36.6	48.2	45.0	39.2	33.2	84.9
+ StreamMeCo	29.1	30.6	26.9	40.3	19.9	28.4	37.0	21.7	36.1	53.0	46.3	41.9	35.1	89.8
+ Random Merging	30.6	31.8	28.8	45.1	19.0	30.4	41.1	21.9	39.6	58.6	41.5	43.2	36.8	94.1
+ ToMe	31.7	27.1	28.6	43.1	20.2	30.6	42.9	23.6	43.6	56.5	49.8	45.7	38.2	97.7
+ MemForest	31.2	**32.9**	**32.4**	43.6	17.4	30.7	**44.8**	23.9	38.9	57.3	50.4	46.3	38.5	98.5
+ **MemFolder (Ours)**	**32.3**	29.4	30.9	**45.4**	19.9	**31.4**	44.5	**28.9**	**44.1**	**60.3**	**51.6**	**48.7**	**40.1**	**102.6**

4.3 Ablation Studies and Analysis

Effects of Each Design. Table 4 evaluates anchor selection in RecordFolder. DPC achieves the best average accuracy of 57.2 and the highest LoCoMo score of 59.3, while MinMax leads on LongMemEval with 55.6. This indicates that progressive density-and-novelty selection balances local representativeness and complementary coverage. Table 5 isolates the assignment signals. The full model averages 57.2; removing cosine similarity causes the largest drop of 3.4 points, compared with 1.9, 1.1, and 0.6 points for gain gating, entity affinity, and temporal chaining. These results identify embedding similarity as the main semantic signal, with the other terms refining entity and temporal consistency. Table 6 compares pruning and merging with DPC anchors fixed. Merging averages 57.2, 6.1 points above pruning and only 0.9 and 0.2 points below the uncompressed scores on LoCoMo and LongMemEval. This mirrors Tables 1 and 2, where merging also performs better under record reduction. Table 7 evaluates ContextFolder after 50% record reduction. The full model achieves 56.6 and 51.8 accuracy with 12.31 and 13.40 tokens per record on LoCoMo and LongMemEval. Removing

Table 4: Anchor selection ablation. Accuracy at 50% record compression.

Settings	LoCoMo	LongMemEval	Average
Random	56.7	53.0	54.8
KMeans	56.5	54.4	55.4
MinMax	58.0	**55.6**	56.8
DPC	**59.3**	55.0	**57.2**

Table 5: Affinity and chaining ablation. Accuracy at 50% record-count reduction.

Settings	LoCoMo	LongMemEval	Average
w/o s_{ent}	58.4	53.8	56.1
w/o cos	55.6	52.0	53.8
w/o gain gate	57.6	53.0	55.3
w/o chaining	59.1	54.2	56.6
Ours	**59.3**	**55.0**	**57.2**

Table 6: Pruning versus merging. Accuracy at 50% reduction with fixed DPC anchors.

Settings	LoCoMo	LongMemEval	Average
Vanilla	60.2	55.2	57.7
Pruning	54.7	47.6	51.1
Merging	**59.3**	**55.0**	**57.2**

Table 7: ContextFolder ablation. Accuracy (%) / tokens per record after 50% record reduction.

Settings	LoCoMo	LongMemEval
w/o planner	55.9 / 16.25	51.0 / 14.44
w/o verifier	54.7 / 11.94	50.4 / 13.19
Ours	**56.6** / 12.31	**51.8** / 13.40

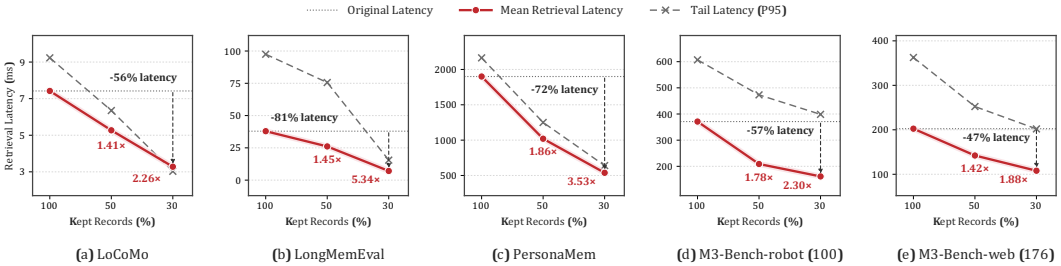


Figure 4: **Retrieval efficiency under record compression.** Mean and 95th-percentile latency with 100%, 50%, and 30% of records retained. Dotted lines indicate uncompressed mean latency. Keeping 30% yields 1.88–5.34× mean speedups across five settings.

the planner lowers accuracy by 0.7 and 0.8 points while increasing token use; removing the verifier saves tokens but costs 1.9 and 1.4 points. The planner thus improves accuracy and compression, while the verifier strengthens recovery at a small token cost. Overall, RecordFolder forms stronger retrieval units, and ContextFolder further compresses their contents.

Efficiency Analysis. Figure 4 reports retrieval latency across three text-based and two multimodal settings. Mean speedups rise from 1.41–1.86× at 50% retention to **1.88–5.34**× at 30%. At 30% retention, mean latency drops by 56%, 81%, and 72% on LoCoMo, LongMemEval, and PersonaMem, respectively; PersonaMem falls from 1,900 to 538 ms. The largest gain appears on LongMemEval, where the mean speedup grows from 1.45× at 50% retention to 5.34× at 30%, while tail latency falls from 97.5 to 15.5 ms. This parallel reduction in mean and tail latency shows that record compression alleviates expensive retrieval cases, not only the average query cost. M3-Bench-robot and M3-Bench-web reach 2.30× and 1.88× speedups, respectively, although their tail-latency gains are smaller than those on the text benchmarks. This gap suggests that record retrieval is only one part of the latency in multimodal settings, where additional processing costs remain. Record compression therefore reduces both average and high-latency retrieval overhead as the external memory grows.

5 Conclusion

Growing external memory stores increase the storage, retrieval, and generation costs of long-term agents. To address these costs, we propose MemFolder, a structure-guided memory folding framework comprising two components: RecordFolder and ContextFolder. The former reduces record count by grouping and consolidating related evidence using semantic, entity, and temporal relations. The latter further reuses these groups for compact encoding, with fact-recovery checks guiding revisions. Experiments across multiple benchmarks demonstrate the strong performance of RecordFolder under record compression. Combined with ContextFolder, existing record compression methods further shorten retrieval units, reducing both record count and context length.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript drafting, language polishing, and structural editing, as well as with literature search and code development.

REFERENCES

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *Proceedings of the International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JroZRaRw7Eu.

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

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

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

Bowen Jiang, Zhuoqun Hao, Young Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo Jose Taylor, and Dan Roth. Know me, respond to me: Benchmarking LLMs for dynamic user profiling and personalized responses at scale. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=6ox8XZGOqP.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, 2023.

Minjae Kim, Jinheon Baek, Soyeong Jeong, and Sung Ju Hwang. Memrefine: Llm-guided compression for long-term agent memory. *arXiv preprint arXiv:2606.13177*, 2026.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, et al. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026a.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, et al. Kimi K3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026b.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 22947–22970, 2024.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, et al. MemOS: A memory OS for AI system. *arXiv preprint arXiv:2507.03724*, 2025.

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

Xuyang Liu, Xiyan Gui, Yuchao Zhang, and Linfeng Zhang. Mixing importance with diversity: Joint optimization for kv cache compression in large vision-language models. In *Proceedings of the International Conference on Learning Representations*, 2026b.

Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory. In *Proceedings of the International Conference on Learning Representations*, volume 2026, pp. 146197–146246, 2026.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 13851–13870, 2024.

James B McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th berkeley symposium on math. stat. and prob.*, pp. 281–297, 1967.

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

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *UIST*, pp. 1–22, 2023.

Tao Ren, Weiyao Luo, Hui Yang, Rongzhi Zhu, Xiang Huang, Yuchuan Wu, Bingxue Chou, Jieping Ye, Jiafeng Liang, Yongbin Li, and Yijie Peng. Scaling self-evolving agents via parametric memory. *arXiv preprint arXiv:2606.04536*, 2026.

Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014. doi: 10.1126/science.1242072.

Junxi Wang, Te Sun, Jiayi Zhu, Junxian Li, Haowen Xu, Zichen Wen, Xuming Hu, Zhiyu Li, and Linfeng Zhang. Streammeco: Long-term agent memory compression for efficient streaming video understanding. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13234–13251, 2026a.

Junxi Wang, Te Sun, Jiayi Zhu, Chen Zhang, Siyuan Li, Xuyang Liu, Zichen Wen, Xiaobing Tu, Jinkui Ren, Xiantao Zhang, et al. Memforest: Efficient agent memory management via eventtree partitioning and progressive merging. *arXiv preprint arXiv:2609.08273*, 2026b.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. MemoryLLM: Towards self-updatable large language models. In *Proceedings of the International Conference on Machine Learning*, pp. 50453–50466, 2024.

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for "important tokens" in multimodal language models: Duplication matters more. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9972–9991, 2025.

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

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 17577–17604, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Atlas Wang, and Beidi Chen. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 34661–34710, 2023.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 19724–19731, 2024.

Jiayi Zhu, Haoxuan Peng, Junxi Wang, Liang Ke, Chen Zhang, and Linfeng Zhang. Large language models do not always need readable language. *arXiv preprint arXiv:2606.19857*, 2026.

APPENDIX

A Additional Experimental Details and Analysis **13**

A.1 Implementation Details 14

A.2 Hyperparameter Analysis 14

B ContextFolder Compression Traces **14**

B.1 Case A: Three Rounds to a Clean Encoding 15

B.2 Case B: One Round on a Redundant Group 16

B.3 Case C: A Substituted Answer Caught by the Judge 16

B.4 On the Reliability of the Check 17

C Prompt Templates **19**

A ADDITIONAL EXPERIMENTAL DETAILS AND ANALYSIS

Memory Framework Details. We consider both text-based and multimodal memory frameworks:

- **Mem0** (Chhikara et al., 2025) is a long-term memory system that extracts, updates, stores, and retrieves persistent textual records from agent interactions. We use Mem0 as the primary text-based framework and construct an uncompressed memory store before applying each compression method.
- **MemOS** (Li et al., 2025) is a memory operating system for AI systems that organizes multiple forms of memory, including plaintext, activation, and parametric memory, under a unified management layer. It provides a framework-level setting for studying record compression beyond a single text-memory implementation.
- **M3-Agent** (Long et al., 2026) is a multimodal agent framework that continuously processes video and audio inputs and builds an entity-centric long-term memory. Its memory integrates visual, auditory, and textual information, which supports multi-turn retrieval and reasoning over dynamic environments.

Benchmark Details. We evaluate long-term memory compression on three conversational benchmarks and two multimodal video benchmarks:

- **LoCoMo** (Maharana et al., 2024) contains long multi-session conversations and questions covering single-hop, multi-hop, temporal, and open-domain reasoning, testing whether an agent can recover evidence distributed across an extended interaction history.
- **LongMemEval** (Wu et al., 2025) evaluates long-term interactive memory with single-session and multi-session questions, preference tracking, knowledge updates, and temporal reasoning.
- **PersonaMem** (Jiang et al., 2025) benchmarks dynamic user profiling and personalized response generation. We use its 32K setting, which tests user-fact recall, preference evolution, recommendation, and generalization across scenarios.
- **M3-Bench-robot** (Long et al., 2026) is an online first-person video benchmark collected from daily robot environments. Its questions require multi-evidence, multi-hop, cross-modal, person-centered, and general-knowledge reasoning over continuously observed interactions.
- **M3-Bench-web** (Long et al., 2026) is an offline web-video benchmark spanning diverse video categories. It evaluates the same long-term multimodal reasoning abilities over prerecorded videos with questions that require evidence from multiple moments and modalities.

Baseline Details. We compare record compression against pruning and merging baselines:

- **Random Pruning** randomly discards records until the target keep ratio is reached. It isolates the effect of record removal without using content, temporal, or graph information.
- **KMeans** (McQueen, 1967) clusters record embeddings into the target number of groups and retains the record closest to each cluster centroid, providing an embedding-space representative-selection baseline.
- **DART** (Wen et al., 2025) is a diversity-based pruning method that removes highly redundant representations by comparing them with a small set of pivot representations. We adapt this operation from visual tokens to memory records.
- **StreamMeCo** (Wang et al., 2026a) compresses structured memory graphs with a dual strategy: it samples representatives from isolated nodes and prunes connected nodes using entity importance and semantic similarity.
- **Random Merging** partitions records into retained and removed sets, then randomly assigns removed records to retained records for consolidation until the target keep ratio is reached.
- **ToMe** (Bolya et al., 2023) progressively matches similar representations across two partitions and merges each matched pair. We adapt this token merging procedure to records by matching each record with its most similar counterpart.
- **MemForest** (Wang et al., 2026b) combines global semantic similarity with local temporal continuity to partition records into event-centric units. It constructs a maximum-spanning-tree EventTree for each unit and progressively merges high-weight pairs, while using anchor-guided propagation to retrieve relevant temporal neighbors.

A.1 Implementation Details

All compression methods start from the same uncompressed memory store built with each memory framework. For Mem0 (Chhikara et al., 2025) and MemOS (Li et al., 2025), we use `gpt-4o-mini` for memory extraction and updating, and `text-embedding-3-small` for record embeddings, following the original papers. For M3Agent (Long et al., 2026), we deploy `gpt-4o` for evaluation and use `text-embedding-3-large` for record embeddings, as in the original paper. The resulting snapshots are shared by all methods before compression. For each target reduction ratio, methods operate on the same records and are evaluated with the same retrieval and question-answering procedure. We preserve the original question sets and report accuracy together with record-count and token-level statistics. Answer quality on Mem0 and MemOS is judged with `gpt-4o-mini`, whereas M3Agent is evaluated with `gpt-4o`. All methods therefore share a common memory and evaluation protocol within each framework.

A.2 Hyperparameter Analysis

We study the sensitivity of RecordFolder's affinity weighting and ContextFolder's iteration limit on LoCoMo and LongMemEval. Figure 5 summarizes the results.

For affinity weighting, the self-adaptive scheme achieves 59.33% on LoCoMo and 55.0% on LongMemEval. The best fixed settings reach 59.09% on LoCoMo ($\alpha = 0.5$ or 0.75) and 53.8% on LongMemEval ($\alpha = 0$), showing a modest gain on LoCoMo and a clearer gain on LongMemEval.

For ContextFolder, three iterations yield the highest accuracy on both benchmarks: 56.56% on LoCoMo and 51.8% on LongMemEval, up from 54.42% and 48.8% with one iteration. At three iterations, token usage is 12.31 and 13.40 tokens per record; across the sweep, it varies only from 11.94 to 12.31 and from 13.08 to 13.43, respectively. More iterations do not improve accuracy or materially reduce token use, so we use three iterations by default.

B ContextFolder Compression Traces

This appendix records three groups compressed end to end on LoCoMo, taken from the production run and reproduced without modification. Each trace lists the source records, the facts and questions the planner produced from them, every encoding the compressor wrote, the blind reader's answers

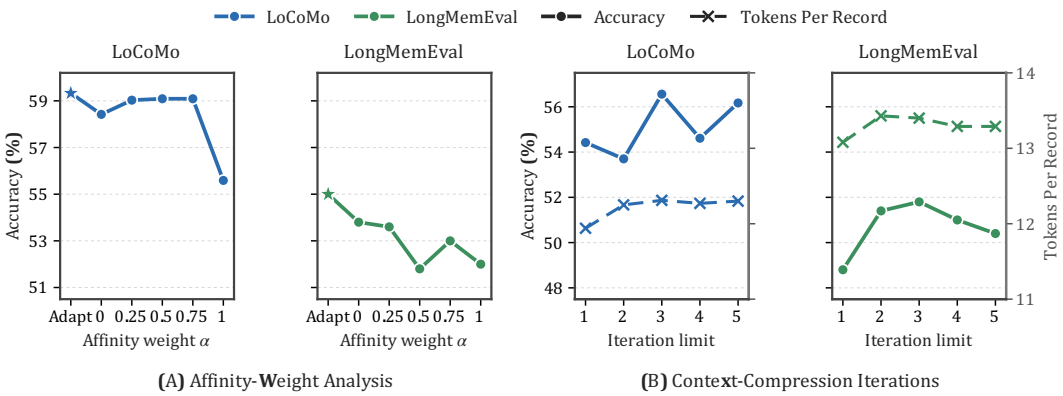


Figure 5: **Hyperparameter analysis on LoCoMo and LongMemEval.** (A) Affinity-weight analysis (panels 1–2); (B) ContextFolder iteration analysis (panels 3–4), each ordered by LoCoMo then LongMemEval. Blue and green denote the benchmarks; solid circles show accuracy and dashed crosses show tokens per record. Higher accuracy and lower token usage are preferred.

and the verifier's verdict on each round, and the encoding finally stored. Token counts are those the implementation itself recorded; the budget of a group is $B_G = \mathrm{round}(\tau \sum_{m \in G} \ell(m))$ with $\tau = 0.8$, and a round is accepted when it loses no required fact and fits within $1.15\, B_G$.

B.1 Case A: Three Rounds to a Clean Encoding

Group `g1s:76433d6a` at $\kappa = 0.7$: four records, 48 tokens, budget 38. The four records say nearly the same thing four times, so the difficulty is not length but deciding which of the overlapping phrasings still have to be recoverable.

#	Record	Tokens
1	User appreciates having people around who accept and support them	11
2	User appreciates their support network for love, guidance, and acceptance	13
3	User appreciates support and is grateful for sharing their journey and helping others	14
4	User appreciates the support of those close to them	10

The planner marked all four facts *high* and *required*, pairing each with a question; fact 4 carried *Who does the user appreciate support from?*, expecting *those close to them*.

Round 1 (19 tokens).

```
E: appreciate.support.network
E: appreciate.support.close
E: grateful.sharing.journey
```

The blind reader recovered facts 1–3 and answered UNKNOWN for fact 4, so the round was returned with one must-fix item:

```
MUST-FIX (required facts the blind reader could not recover):
  * User appreciates the support of those close to them.
    Express THESE explicitly; clarity first for them.
    Re-encode by EDITING the previous encoding. Restore ONLY the facts
    listed as MUST-FIX; do not spend tokens on optional facts unless you
    are well under budget; if over budget, tighten wording or drop
    optional facts.
```

Round 2 (26 tokens). The compressor edited its previous encoding in place, adding `E: appreciate.support.acceptance`. Fact 4 was still answered UNKNOWN, and the same item was returned a second time.

Round 3 (21 tokens).

```
E: appreciate.support.acceptance
E: appreciate.support.love.guidance
E: appreciate.support.close
```

All four questions were answered correctly, weighted recovery reached 1.0, and $21 \leq 1.15 \times 38$, so the loop stopped early.

Stored. Round 3, at 21 tokens and a ratio of 0.44. Round 1 was two tokens shorter and was not eligible: it lost a required fact, so it never entered $\mathcal{C}$. This is what the subscript in Eq. (1) does — the stored encoding is the shortest among the rounds that passed, which need not be the shortest round.

B.2 Case B: One Round on a Redundant Group

Group `g1s:2c6ea7c5` at $\kappa = 0.7$: six records, 62 tokens, budget 50. Not every group needs the loop. Here each of three facts is stated exactly twice, and the planner's triage removes the duplication before the compressor writes anything.

#	Record
1	Volunteering is an important part of Caroline's life
2	Caroline is carrying on volunteering at the youth center
3	Volunteering at the youth center is an important part of Caroline's life
4	Caroline wants to keep making a difference
5	Caroline is carrying on volunteering at the youth center
6	Caroline wants to keep making a difference

The planner returned three facts: volunteering matters to her *(high)*, she volunteers at the youth center *(high)*, and she wants to keep making a difference *(med)*.

Round 1 (17 tokens), accepted.

```
E: volunteering important
A -> volunteering at youth center
? keep making a difference
```

Every question was answered correctly on the first attempt, giving a ratio of 0.27. Re-running this group reproduces the cached encoding byte for byte.

B.3 Case C: A Substituted Answer Caught by the Judge

Group `g1s:074d8b3e` at $\kappa = 0.3$: twelve records, 92 tokens, budget 74. This is the group cited in the ContextFolder section. Records 4 and 5, 6 and 7, and 9 and 10 are near-duplicates, and timestamps are excluded from the fact list by rule, so the planner reduced twelve records to six facts.

#	Record	Date
1	Cares about mental health	27 Jun 2023
2	User is keen on counseling or working in mental health	27 Jun 2023
3	Values counseling and support groups	27 Jun 2023
4	Thinking about taking care of mental health	25 May 2023
5	Values taking care of mental health	25 May 2023
6	Interested in counseling and mental health	3 Jul 2023
7	Keen on counseling or working in mental health	3 Jul 2023
8	Looking into counseling and mental health jobs	12 Jul 2023
9	Looking into counseling and mental health as a career	12 Jul 2023
10	Looking into counseling and mental health as a career	12 Jul 2023
11	Started looking into counseling and mental health career options to help others	12 Jul 2023
12	Values sharing stories and giving others hope	9 Jun 2023

#	Fact (level)	Question → expected answer
1	Cares about mental health *(high)*	What does the user care about? → Mental health
2	Keen on counseling or working in mental health *(high)*	Interest regarding career? → Counseling or mental health work
3	Values counseling and support groups *(med)*	What does the user value? → Counseling and support groups
4	Looking into counseling and mental health jobs *(high)*	Currently exploring? → Counseling and mental health jobs
5	Started looking into these options to help others *(med)*	Why? → To help others
6	Values sharing stories and giving others hope *(med)*	Values in support? → Sharing stories, giving hope

Round 1 (44 tokens). The compressor came in well under budget by writing five lines and omitting facts 3 and 6.

```
cares about mental health
keen on counseling or working in mental health
looking into counseling and mental health jobs
started looking into counseling and mental health career options to help others
values sharing stories and giving others hope
```

Questions 1, 2, 4 and 5 were answered correctly. Question 3 asked what the user valued in relation to mental health; the encoding no longer carried fact 3, and the blind reader answered *sharing stories and giving others hope* — a fact the encoding did carry, offered in place of the one that was lost. Nothing in that answer looks like a failure on its own, and a check that only watched for UNKNOWN would have passed it; the judging call compared it against the planner's expected answer and marked it must-fix. Question 6 was answered UNKNOWN. Both items were returned to the compressor:

```
MUST-FIX (required facts the blind reader could not recover):
  * User values counseling and support groups.
  * User values sharing stories and giving others hope.
    Express THESE explicitly; clarity first for them.
    Re-encode by EDITING the previous encoding. Restore ONLY the facts
    listed as MUST-FIX; do not spend tokens on optional facts unless you
    are well under budget; if over budget, tighten wording or drop
    optional facts.
```

Round 2 (50 tokens), accepted.

```
cares about mental health
keen on counseling or working in mental health
values counseling and support groups
looking into counseling and mental health jobs
started looking into counseling and mental health career options to help others
values sharing stories and giving others hope
```

All six questions were recovered and the loop stopped early, at a ratio of 0.54.

B.4 On the Reliability of the Check

The blind check is stochastic in both directions, as traces show. In Case C, the line `values sharing stories and giving others hope` appears verbatim in the round-1 encoding, yet fact 6 was answered UNKNOWN; the line the compressor added in round 2 to repair it is the line that was already there, and attempt 2 passed. In Case A, the line `E: appreciate.support.close` appears in rounds 1 and 3 and was judged differently each time.

We report this instead of tuning it away, because the two directions do not cost the same thing. A fact the encoding does carry but the reader fails to recover costs the loop a round of compute, and the encoding that is eventually stored still passed a check; a fact the encoding has genuinely lost but the reader appears to recover would cost fidelity, and is the failure this design is built to avoid.

Requiring recovery by a reader who cannot see the records makes the check err towards the first, and $\mathcal{C}$ in Eq. (1) is defined over rounds that passed it.

C PROMPT TEMPLATES

We provide the prompts used by the main LLM components in execution order.

Entity Grounding Prompt: Initial Session

```
Role: exhaustive entity extraction for a memory compression graph.

You will ground every memory node of one conversation session. The extracted
entities decide which memories are allowed to merge, so recall of
content-bearing entities matters more than precision of wording.

Session metadata:
- sample_id: {sample_id}
- exact_timestamp: {timestamp}

Extraction rules:
1. Extract ALL content-bearing entities mentioned or clearly implied by the
   node text: people (including authors/creators/ public figures), named works
   (books, movies, songs, shows) together with their creators, places and
   locations, organizations / communities / groups, pets or named animals,
   significant objects / possessions, activities / hobbies / jobs /
   fields of work, topics or concepts someone could ask about later, and
   concrete events.
2. The bare word "User" refers to the source speaker; resolve it to the
   speaker's own name and do not emit "user" as an entity.
3. Do NOT stop at the protagonist. A node like "Loved 'X' by Y" must yield
   both the work X and the person Y. Aim for at least 2 entities per node
   whenever the text carries any concrete content; only pure greetings or
   chit-chat may have fewer.
4. `canonical` must be a short noun phrase in lowercase, without a leading
   "the", without verbs. Aliases of the same thing share one canonical label.
5. Do not invent entities that are not in the text.
6. `relations` are optional; keep them only when clean, with a noun-phrase
   subject and object (never a verb phrase or sentence).
7. Set `opens_channel=true` for any entity that could plausibly recur in
   later memories.

Allowed entity types: {types}

Return exactly one JSON object:
{
  "nodes": [
    {
      "id": 0,
      "entities": [
        {"canonical": "becoming nicole", "type": "work", "roles": ["liked"], "mentions":
     ["Becoming Nicole"], "opens_channel": true},
        {"canonical": "amy ellis nutt", "type": "person", "roles": ["author"], "mentions":
      ["Amy Ellis Nutt"], "opens_channel": true}
      ],
      "relations": [{"subject": "amy ellis nutt", "predicate": "wrote", "object": "
     becoming nicole"}]
    }
  ]
}

Every input id must occur exactly once. Output JSON only, no commentary.

INPUT NODES:
{records}
```

Figure 6: **Initial-session entity grounding prompt.** The prompt extracts content-bearing entities from one conversation session without an earlier canonical-label registry.

Entity Grounding Prompt: Cross-Session Registry

```
Role: exhaustive entity extraction for a memory compression graph.

You will ground every memory node of one conversation session. The extracted
entities decide which memories are allowed to merge, so recall of
content-bearing entities matters more than precision of wording.

Session metadata:
- sample_id: {sample_id}
- exact_timestamp: {timestamp}
- known/recurrent canonical labels from earlier sessions: {registry}

Extraction rules:
1. Extract ALL content-bearing entities mentioned or clearly implied by the
   node text: people (including authors/creators/public figures), named works
   (books, movies, songs, shows) together with their creators, places and
   locations, organizations / communities / groups, pets or named animals,
   significant objects / possessions, activities / hobbies / jobs /
   fields of work, topics or concepts someone could ask about later, and
   concrete events.
2. The bare word "User" refers to the source speaker; resolve it to the
   speaker's own name and do not emit "user" as an entity.
3. Do NOT stop at the protagonist. A node like "Loved 'X' by Y" must yield
   both the work X and the person Y. Aim for at least 2 entities per node
   whenever the text carries any concrete content; only pure greetings or
   chit-chat may have fewer.
4. `canonical` must be a short noun phrase in lowercase, without a leading
   "the", without verbs. Aliases of the same thing share one canonical label.
5. Coreference discipline:
   a. If an entity is the SAME referent as a known/recurrent label above,
      reuse that label EXACTLY.
   b. If it merely shares words but is a DIFFERENT referent (e.g. a different
      person with the same first name, different kids, a different song),
      create a disambiguated label, e.g. "max (neighbor)" vs "max (coworker)".
   c. Generic referents must carry owner qualification when an owner is
      known: "melanie's kids", not bare "kids"; "caroline's garden", not
      "garden". Truly public/generic concepts (art, pottery, mental health)
      stay unqualified.
6. Do not invent entities that are not in the text.
7. `relations` are optional; keep them only when clean, with a noun-phrase
   subject and object (never a verb phrase or sentence).
8. Set `opens_channel=true` for any entity that could plausibly recur in
   later memories.

Allowed entity types: {types}

Return exactly one JSON object:
{
  "nodes": [
    {
      "id": 0,
      "entities": [
        {"canonical": "becoming nicole", "type": "work", "roles": ["liked"], "mentions":
     ["Becoming Nicole"], "opens_channel": true},
        {"canonical": "amy ellis nutt", "type": "person", "roles": ["author"], "mentions":
      ["Amy Ellis Nutt"], "opens_channel": true}
      ],
      "relations": [{"subject": "amy ellis nutt", "predicate": "wrote", "object": "
     becoming nicole"}]
    }
  ]
}

Every input id must occur exactly once. Output JSON only, no commentary.

INPUT NODES:
{records}
```

Figure 7: **Cross-session entity grounding prompt.** The prompt adds canonical-label reuse and owner-qualified coreference resolution.

Memory Merge Prompt

```
You are an intelligent memory summarization assistant. Please extract the core information
     from the following {N} memory entries and their timestamps, and
generate a concise and coherent summary paragraph. If the memories contain relative time
    expressions (e.g., "last June," "two months ago"), convert them into
specific dates based on the corresponding timestamps (for example, if a memory mentions "
    last June" and the timestamp is February 2023, the actual time should be
June 2022). The summary length should be kept within the combined length of the original
    memories as much as possible. Output only the final summarized result,
without any explanations, analysis, reasoning steps, or additional content.
  Memory 1: {text}, Timestamp 1: {ts}
  ...
  Memory N: {text}, Timestamp N: {ts}
```

Figure 8: **Memory merge prompt.** The LLM consolidates records assigned to one RecordFolder group and resolves relative dates from timestamps.

ContextFolder Planner Prompt

```
From the memory entries below, extract the KEY facts as a numbered list, and for each fact
     write a short verification question with its expected answer. A
reader should be able to answer the question IF AND ONLY IF the fact is recoverable from a
     compressed encoding.

Scale the fact count to the input size: about 1 atomic fact per 10 source tokens (min 2,
    max 12), one question per atomic fact. A short single memory deserves
only 2-4 questions about its ONE core fact (e.g. who/what/when), not a decomposition into
    trivia. For each fact also provide:
  - "importance": "high" for key events, final states, exact dates/numbers/names, entity
    bindings, negations; "med" for preferences, plans, relation directions,
    quantities; "low" for reasons, modifiers, emotional framing, superseded older states
  - "inferable": true if a reader could infer this fact from the OTHER extracted facts or
    from common knowledge, else false
Do NOT include meta information such as the memory entries' own timestamps as facts. Keep
    each fact atomic.

Memories:
{lines}

Answer with JSON only: {"facts": [{"fact": "...", "question": "...", "answer": "...", "
    importance": "high|med|low", "inferable": true|false}]}
```

Figure 9: **Planner prompt.** The planner extracts atomic facts and creates blind verification questions without access to downstream test queries.

ContextFolder Compressor Prompt

```
You are a memory compressor writing for another equally intelligent LLM, not for humans.
     Compress the {n} memory entries below into a compact token sequence
that another LLM can decode WITHOUT any external codebook.

Fixed grammar (one fact per line):
- `@YYYY-MM-DD` ONLY when a fact itself contains time information; NEVER attach entry
     timestamps to timeless facts
- `A -> B` for state changes; `E: state` for attributes; `~` negation; `?` uncertain
- Exact entity full names; exact numbers/dates verbatim
- NEVER add `User:`/`She:` prefixes unless the source itself names the subject
- cl100k discipline: dotted fragments like `M.son.acc` are EXPENSIVE, never use them;
     common whole words are cheap
- Short single memories must come out strictly shorter than the source: keep verb + object
     , drop titles' authors, reasons, feelings. E.g. "Loved 'Becoming
Nicole' by Amy Ellis Nutt" -> `loved 'Becoming Nicole'`

SOFT TOKEN BUDGET: about {target} tokens (source: {src}).

TRIAGE THE FACTS BEFORE ENCODING. The key facts are listed below with labels:
- MUST KEEP: [high] facts that are NOT inferable  key events, final states, exact names/
     dates/numbers, negations. These must survive verbatim-recoverable.
- DROP FIRST: facts marked [inferable]  the reader can re-derive them from the facts you
     keep or from common knowledge. Do not spend a single token on them.
- DROP IF TIGHT: [low] facts  reasons, modifiers, emotional framing, superseded older
     states.
- [med] facts: keep only while you are within budget.
Omitting a fact is allowed and expected; distorting or inventing one is not.

Key facts to triage:
{facts_list}

Output ONLY the encoding.

Memories:
{lines}
```

Figure 10: **Compressor prompt.** The compressor writes a compact encoding under a soft token budget and prioritizes non-inferable facts.

ContextFolder Verifier: Blind Answerer

```
You see ONLY a compact memory encoding and a list of questions. The encoding uses this
     grammar: `@YYYY-MM-DD` = the fact happened on that date; `A -> B` = A
changed into / transferred to B; `E: x` = entity E has attribute/state x; `~` = negation;
     `?` = uncertain/rumored; `;` and newlines separate facts; filler words
are dropped but entity names, numbers, dates and relation directions are preserved. First
     silently decode each line into plain facts, then answer each question
using ONLY what can be inferred from the decoded facts. If a question cannot be answered
     from the encoding, answer exactly "UNKNOWN". Do not guess.

Encoding:
{enc}

Questions:
{questions}

Answer with JSON only: {"answers": ["...", ...]} with exactly {k} entries.
```

Figure 11: **Blind-answerer prompt.** The verifier tests whether a reader can recover planner facts from the encoding alone.

ContextFolder Verifier: Fact Judge

```
Reference key facts with expected answers:
{facts}

A blind reader who saw only the compressed encoding answered the verification questions as
      follows:
{answers}

For each item, decide whether the fact is recoverable from the encoding. Be LENIENT:
     accept semantically close or partial answers, accept answers that are
reasonably inferable from the encoding even if wording differs; only exact numbers/dates/
     names must match. Mark false only when the answer is missing, "UNKNOWN",
or contradicts the fact.
Answer with JSON only: {"ok": [true, false, ...]} with exactly {k} entries.
```

Figure 12: **Fact-judge prompt.** The judge evaluates recoverability from the blind reader's answers and uses lenient semantic matching.

ContextFolder Revision Feedback Prompt

```
VERIFIER FEEDBACK on your previous attempt ({prev_tokens} tokens, budget {target}):
---
{prev}
---
{issues}
Re-encode by EDITING the previous encoding. Restore ONLY the facts listed as MUST-FIX; do
     not spend tokens on optional facts unless you are well under budget;
if over budget, tighten wording or drop optional facts. Output ONLY the new encoding.

The rule-generated issue block has one or more of the following forms:

MUST-FIX (required facts the blind reader could not recover):
- {fact}
...
Express THESE explicitly; clarity first for them.

OVER BUDGET: {ntok} vs {target} tokens. Tighten wording, merge lines, or drop the {n}
     optional facts entirely.

Note: {n} optional (low/inferable) facts were dropped; that is acceptable.
```

Figure 13: **Revision feedback prompt.** The feedback is appended to later compression rounds and contains only rule-generated revision signals.

ContextFolder Degenerate Fallback Prompt

```
You are an intelligent memory summarization assistant. Please extract the core information
      from the following {n} memory entries and their timestamps, and
generate a concise and coherent summary paragraph. If the memories contain relative time
     expressions, convert them into specific dates based on the corresponding
timestamps. The summary length should be kept within the combined length of the original
     memories as much as possible. Output only the final summarized result,
without any explanations.
  {lines}
```

Figure 14: **Degenerate fallback prompt.** This one-shot summarizer handles groups for which the planner does not return usable facts.

Memory Question-Answering Prompt

```
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.
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 5-6 words.

# APPROACH (Think step by step):
1. First, examine all memories that contain information related to the question
2. Examine the timestamps and content of these memories carefully
3. Look for explicit mentions of dates, times, locations, or events that answer the
     question
4. If the answer requires calculation (e.g., converting relative time references), show
     your work
5. Formulate a precise, concise answer based solely on the evidence in the memories
6. Double-check that your answer directly addresses the question asked
7. Ensure your final answer is specific and avoids vague time references

Memories for user {{speaker_1_user_id}}:

{{speaker_1_memories}}

Memories for user {{speaker_2_user_id}}:

{{speaker_2_memories}}

Question: {{question}}

Answer:
```

Figure 15: **Memory QA prompt.** The prompt answers benchmark questions from the retrieved memories and their timestamps.

Compact-Encoding Decode Legend

```
# IMPORTANT - COMPACT ENCODED MEMORIES:
Some memories are stored in a compact machine-oriented encoding rather than plain
English. BEFORE answering, silently decode each memory into the facts it encodes.
Decoding legend: `@YYYY-MM-DD` = the fact happened on that date; `A -> B` = A
changed into / transferred to B; `E: x` = entity E has attribute or state x;
`~` = negation; `?` = uncertain or rumored; `;` and newlines separate facts;
compact wording drops filler words but always preserves entity names, exact
numbers, dates and relation directions. First expand each encoded line into its
plain-English facts, then answer strictly from the decoded facts.
```

Figure 16: **Decode legend for compact memories.** This legend is injected before the QA instructions when the retrieved records use ContextFolder encodings.

Answer Accuracy Judge Prompt

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

The point of the question is to ask about something one user should know about the other
     user based on their prior conversations.
The gold answer will usually be a concise and short answer that includes the referenced
     topic, for example:
Question: Do you remember what I got the last time I went to Hawaii?
Gold answer: A shell necklace
The generated answer might be much longer, but you should be generous with your grading -
     as long as it touches on the same topic as the gold answer, it should
be counted as CORRECT.

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

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

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

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

Figure 17: **Answer-accuracy judge prompt.** The judge labels a generated answer against the benchmark question and gold answer.