3D-MEM: LEARNING TO ORGANIZE EVIDENCE FOR LONG-HORIZON CONVERSATIONAL MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

A long-horizon conversational agent can retain the information needed to answer a question yet fail to recover it within a compact context. Relevant details may remain buried in coarse summaries, disconnected across sessions, or obscured by redundant memories. We address this gap between information retention and evidence access with **3D-Mem**, a framework that learns how to organize conversational evidence for future retrieval. Its central mechanism couples task-feedback-guided graph-operator induction with a provenance-linked evidence index: a PPO-trained controller selects graph edits, an LLM designer refines the operator library from training failures, and the resulting topology supplies retrieval signals grounded in source segments. Hierarchical reranking projects these signals back to the supporting text, while utility gating and diversity pruning assemble the answer context. Trained on LoCoMo with Qwen, the controller and operator library are frozen for evaluation across two benchmarks and two backbones. On LoCoMo with Qwen, 3D-Mem improves overall F1 over A-Mem by 5.58 points. With Qwen, it uses 24.2% fewer retrieved tokens than Mem0; holding graph architecture and PPO budget fixed, freezing the initial operator library lowers F1 by 1.11 points. These results support adapting memory organization to downstream evidence needs.

1 INTRODUCTION

A conversational agent may have encountered the answer and still be unable to recover it. Across a long interaction history, a specific event can be absorbed into a broad summary, references to the same entity can be scattered across sessions, and repeated mentions can crowd out the detail a question requires. Retaining more history alone does not ensure that the right evidence reaches the answer generator: even long-context models can struggle to use relevant information embedded in lengthy inputs (Liu et al., 2024). For external memory, this motivates a distinction between *information retention* and *evidence access*: whether stored information can be located at the required granularity and brought together within a limited retrieval budget.

Evidence organization also affects how retrieved facts are used. In our case study (Appendix D), a retrieved memory already names `Ferrari`, yet the agent answers "new sports car". A later event-level memory explicitly connects the purchase, the entity, and the query's descriptive phrase, and the answer becomes specific. This qualitative example motivates a question about memory construction: *how should an agent organize experience so that future questions can locate and use the evidence they need?*

Existing work provides important parts of the answer. Hierarchical and graph memories organize experience across temporal scales and relational views (Li et al., 2026; 2025; Jiang et al., 2026); reinforcement learning adapts memory-management decisions (Yan et al., 2025; Wang et al., 2025); and MemSkill learns to select and refine reusable memory operations (Zhang et al., 2026). Building on these advances, we study how task feedback can shape a *source-grounded evidence index*. This requires connecting three steps: changing how entities and events are represented, translating structural relevance into support for the original evidence, and selecting complementary evidence for a compact answer context.

We propose **3D-Mem**, a framework that learns to organize evidence through three decisions: evolution, retrieval, and assembly. A multi-scale temporal hierarchy stores conversational evidence, while

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.

a semantic graph provides relational access through anchors to source segments. During training, a PPO controller selects graph-operation skills and an LLM designer refines the operator library from QA failures. The induced operators specify edit scopes and provenance requirements, changing how the graph represents and connects evidence. At retrieval time, relevant graph elements contribute scores to their anchored source segments; hierarchical reranking combines this support with local similarity and branch-level evidence. Utility gating and diversity pruning then reduce irrelevant and repeated content.

The key design is the connection from **task feedback, through graph edits, to source-evidence selection**. Because graph relevance contributes directly to source-chunk scores, changing the graph can change which evidence reaches the answer generator. The controller and operator library are trained on LoCoMo with Qwen and then frozen for evaluation; retrieval uses a specified scoring rule, and assembly uses a prompted gate and fixed diversity threshold. Thus, learned memory construction supports evidence access without requiring test-time operator induction.

Our contributions are:

- **Task-adaptive evidence organization.** We develop graph-operation skills with explicit edit scopes and source anchors, connecting feedback-driven operator induction to the construction of a semantic evidence index.
- **Source-grounded use of the evolving index.** We connect graph relevance to source segments through provenance projection and hierarchical scoring, then assemble complementary evidence across memory views.
- **Evaluation of quality, context cost, and adaptation.** Experiments cover two benchmarks and two backbones. Controlled ablations examine operator evolution and graph-to-source projection with stored memory held fixed for the latter, alongside context assembly.

2 Related Work

External-memory systems range from vector stores such as RAG and MemGPT (Lewis et al., 2020; Packer et al., 2023) to hierarchical, temporal, and graph memories (Li et al., 2026; 2025; Han et al., 2025; Jiang et al., 2026). MAGMA uses multiple relational graphs for query-adaptive traversal and context construction. Memory-R1 and Mem-α learn memory-management decisions, while NEMORI adapts memory retention (Yan et al., 2025; Wang et al., 2025; Ma et al., 2026). Mem-Skill learns skill selection and uses an LLM designer to evolve memory operations from hard cases (Zhang et al., 2026). 3D-Mem builds on this controller–designer organization to induce graph edits with explicit source anchors. These anchors also transmit graph relevance to source-chunk scores, coupling operator evolution to the evidence selected downstream. This makes the construction-time edit schema part of the retrieval interface: the relations built by the induced operators provide source-grounded signals for evidence selection. Appendix A provides further discussion.

3 Method

3.1 Overview: Three Decisions Across the Memory Lifecycle

3D-Mem couples a dual-axis memory representation with three decisions governing its construction and use. Task feedback shapes the graph-operation library; the selected operators modify a provenance-linked topology that guides evidence retrieval. Utility gating and diversity pruning then assemble the retrieved evidence into the final reasoning context.

Dual-Axis Memory Representation Space (§3.2). As the decision substrate of 3D-Mem, the Dual-Axis Memory Representation Space comprises a Multi-Scale Temporal Axis ($\mathcal{T}$) organizing evidence across temporal scales, and a Semantic Topology Axis ($\mathcal{G}$) maintained by a dynamic knowledge graph.

Decisions for Evidence Organization and Access. Operating on this substrate, 3D-Mem coordinates three mechanisms across the memory lifecycle. (1) **Evolution Decision (§3.3):** a dual-encoder

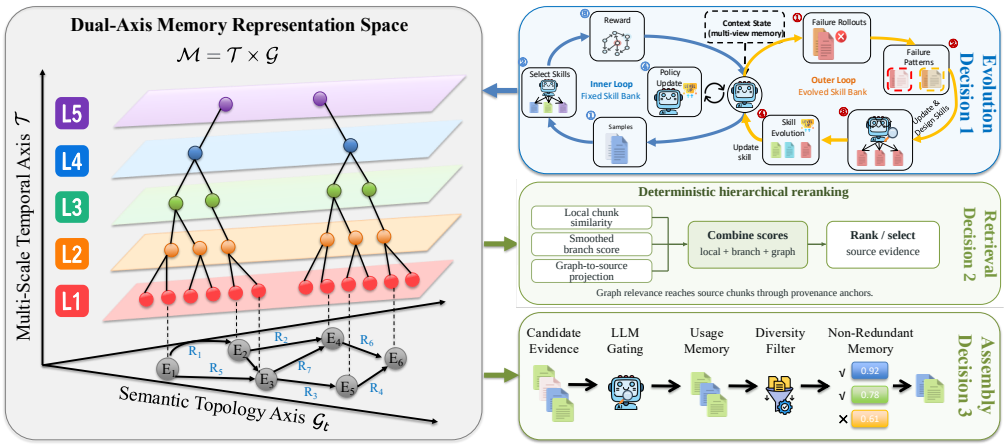


Figure 1: 3D-Mem organizes conversational memory through evolution, retrieval, and assembly. Retrieval combines local similarity, smoothed branch scores, and graph relevance projected through source anchors; its scoring rule is fixed.

policy and an outer-loop skill updater jointly evolve graph topology and the operation space. (2) **Retrieval Decision (§3.4):** calibrated hierarchical reranking combines branch-level evidence support with provenance-projected graph relevance. (3) **Assembly Decision (§3.5):** utility gating and diversity pruning select useful evidence and filter cross-source redundancy.

3.2 Dual-Axis Memory Representation Space

We define the underlying state space of the agent's memory as a Cartesian product $\mathcal{M} = \mathcal{T} \times \mathcal{G}$, spanned by a multi-scale temporal hierarchy and a semantic topology. Inspired by memory consolidation (McClelland et al., 1995), the temporal axis groups evidence at increasing time scales, while the semantic axis links related entities and events across sessions.

Multi-Scale Temporal Axis. The temporal axis $\mathcal{T}$ is a multi-scale hierarchy comprising five representation layers $\{L_k\}_{k=1}^{5}$. These layers span increasingly broad time scales and levels of abstraction, from fine-grained factual records to behavioral patterns and persistent preferences. As summarized in Table 1, each layer plays a distinct role in evidence organization and retrieval.

Table 1: Representation levels of the 3D-Mem multi-scale temporal axis.

Layer	Span	Representation	Content & Utility
L_1	Minutes	Factual Segment	Retains fine-grained dialogue evidence for factual and chronological queries.
L_2	Session	Event Summary	Aggregates core facts within a session through prompted summarization.
L_3	Daily	Daily Pattern	Summarizes recurring activities and routines across sessions within a day.
L_4	Weekly	Behavioral Trait	Summarizes preferences, habits, and behavioral changes across days.
L_5	Months+	Structured Profile	Consolidates identity, interests, preferences, and salient events across longer histories.

Each higher-level memory is generated from lower-level records and its previous state:

$$L_k = \Phi_k\big(C_k, H_k, I_k\big) \tag{1}$$

where $C_k = \{L_{k-1}^{(i)}\}_{i \in \mathcal{W}_k}$ represents the constituent sub-memories within the sliding temporal window $\mathcal{W}_k$, H_k denotes the historical memory at the same layer providing semantic continuity, and I_k stands for the layer-specific instruction guiding consolidation.

Layer-specific instructions I_k encourage summaries to retain salient facts and express broader patterns compactly. This is a prompt-level design objective: we do not estimate an information-theoretic density or enforce a monotonic density constraint. Appendix B provides the prompts. The L_1 records remain available to recover details omitted by higher-level summaries.

Semantic Topology Axis. The semantic axis $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ is a dynamic knowledge graph that captures semantic relations beyond the linear temporal order. It is bidirectionally linked to the temporal axis: each entity or relation is grounded in L_1 factual segments, while each segment records the semantic elements it contains. This design enables the agent to connect abstract semantic structure with chronological evidence for cross-axis retrieval.

3.3 Evolution Decision: Co-evolution of Graph Structure and Memory Skills

We update the semantic graph as sessions arrive using a dual-loop mechanism. The inner loop selects and applies provenance-preserving operations to update the graph $\mathcal{G}$, and the outer loop adds, revises, or removes operations in the skill library $\mathcal{A}$ from task feedback. Thus the system evolves both objects jointly,

$$(\mathcal{G}_k, \mathcal{A}_k) \longrightarrow (\mathcal{G}_{k+1}, \mathcal{A}_{k+1}), \tag{2}$$

where $\mathcal{A}_k$ is the library of graph-operation skills at outer iteration k. Each skill specifies a target view, readable source views, an edit scope, permitted graph actions, and execution instructions. The library evolves over a shared set of primitive edit types, such as entity insertion, entity merging, and relation updates. Graph writes require provenance anchors to supporting L_1 chunks. These constraints connect induced operators to the memory schema and source evidence; the resulting edits change the graph support available to source-evidence retrieval. Appendix C and the execution prompts in Appendix E provide the schemas.

3.3.1 Inner Loop: Policy-Driven Graph Evolution

We formulate knowledge-graph maintenance as an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. At step t, the state

$$s_t = \psi(\text{session}_t) \oplus \phi(\mathcal{R}_t^{L_1}) \oplus \phi(\mathcal{R}_t^{L_2}) \oplus \phi(\mathcal{R}_t^{\mathcal{G}}) \tag{3}$$

combines the current session with retrieved L_1 segments, L_2 summaries, and the semantic subgraph. Here $\psi(\cdot)$ is a pre-trained language-model encoder and $\phi(\cdot)$ encodes each memory source.

To accommodate the evolving library $\mathcal{A}$ (§3.3.2), a dual-encoder policy $\pi_\theta(a \mid s_t)$ scores each candidate skill $a_i \in \mathcal{A}$ by

$$z_{t,i} = \text{MLP}_{\text{inter}}\big(s_t \oplus \psi_{\text{skill}}(a_i)\big), \tag{4}$$

where $\psi_{\text{skill}}(a_i)$ encodes the instruction template of skill a_i. Scores $\mathbf{z}_t/\tau$ parameterize Gumbel-Top-k selection of three skills; the LLM executor then generates graph actions conditioned on those skills. Appendix C provides the initial library, and Appendix E.2 specifies the training procedure.

The training reward combines conversation-level QA F1 with a step-level shaping reward for consistency between selected skills and executor actions. Delayed QA reward is distributed across session-level decisions, and the policy is optimized with GAE and clipped PPO (Schulman et al., 2017). Appendix E.2 specifies the rollout and credit-assignment procedure.

3.3.2 Outer Loop: Task-Feedback-Guided Graph-Operator Induction

At the end of each training phase, a Meta-Reflector collects failed QA trajectories and clusters their query representations with K-means. The grouped failures and retrieved memories provide context for identifying recurring construction issues, such as redundant entities, missing relations, or overly coarse event representations.

A frozen LLM designer receives these failures, the current operator schemas, and the evolution history. It proposes additions (ADD_SKILL), template refinements (REFINE_TEMPLATE), or removals (DROP_SKILL) within the graph-edit schema. Each induced operator specifies when and how to apply its permitted edits using the available source views. The controller subsequently selects among the updated operators during graph construction, allowing task feedback to reshape the evidence index accessed by future queries.

3.4 Retrieval Decision: Evidence-Count-Calibrated Hierarchical Reranking

At query time, we rank source evidence using local semantic similarity, branch-level support, and graph provenance. This deterministic scoring procedure requires no LLM call; the prompted utility gate is applied later during Assembly.

3.4.1 Evidence-Count-Aware Additive Smoothing

Let $r_c = \cos(e_c, e_q) \in [-1, 1]$ denote the raw cosine similarity between chunk c and query q. Before applying the hierarchical score, we use the affine transformation

$$s_c = \frac{r_c + 1}{2} \in [0, 1]. \tag{5}$$

We do not clip negative cosine similarities; they are mapped continuously below 0.5. For a non-leaf memory node m with descendant chunks $\mathrm{Desc}(m) \subseteq L_1$, we compute

$$\bar{s}_m = \frac{1 + \sum_{c \in \mathrm{Desc}(m)} s_c}{2 + |\mathrm{Desc}(m)|}. \tag{6}$$

Equation 6 applies evidence-count-aware additive smoothing to distinguish branches with the same empirical mean but different amounts of support. Estimates based on few descendants shrink toward 0.5, while $\bar{s}_m$ approaches the empirical mean as the number of descendants grows. The resulting score supplies branch context for provenance-aware reranking. Appendix E.3 provides the scoring configuration and pseudo-count formulation.

3.4.2 Unified Reranking with Graph Structure

The semantic graph acts as an evidence index: query-relevant graph elements contribute support to their corresponding L_1 chunks through provenance anchors. This projection connects the topology produced by graph operators to the source evidence used for answering. Combining graph support $\Delta_{\mathcal{G}}$ with local similarity and the smoothed branch score gives

$$\mathrm{Score}(c) = \omega_1 \cdot s_c + \omega_2 \cdot \bar{s}_{\mathrm{parent}(c)} + \Delta_{\mathcal{G}}(c), \tag{7}$$

where $\Delta_{\mathcal{G}}(c) = w_g \cdot \mathrm{mean}\big(\mathrm{top}_M\{\mathrm{score}(a) \mid a \in \mathrm{Support}(c)\}\big)$ aggregates support from the top-M relevant relation nodes anchored to c.

This scoring function enables the agent to retrieve evidence beyond naive vector similarity by combining local relevance with contextual and structural support. Starting from the base similarity score s_c, the other two terms adjust retrieval in complementary ways:

- *Branch context:* Chunks in more relevant branches receive a larger contextual contribution; additive smoothing reduces the deviation from 0.5 for branches with few descendants.
- *Graph-to-source support:* Relevant relation nodes increase the scores of their anchored source chunks, providing a signal beyond direct query–chunk similarity.

3.5 Assembly Decision: Diversity-Aware Redundancy Elimination

The retrieved candidate set may contain two kinds of redundancy: evidence that is relevant but unhelpful for the current query, and cross-source overlap where different memory axes express the same fact. To improve prompt efficiency, we use a two-stage refinement pipeline: a prompted LLM utility gate followed by diversity pruning.

Prompted Utility Gating. Not all retrieved segments are equally useful for answering the query. A prompted LLM gate assesses whether each candidate contains facts or reasoning cues useful for the question and selects candidates for subsequent filtering.

Embedding-based Diversity Pruning. After utility gating, we further remove semantic overlap across sources, such as redundancy between graph triplets and their supporting raw text chunks. We apply a greedy similarity filter using the dense representations v precomputed during indexing. The

Table 2: Results on the LoCoMo test set. F1 and L-J (GPT-4o-mini) are percentages. Best and second-best results are boldfaced and underlined.

Method	Single-Hop		Multi-Hop		Temporal		Open-Domain		Overall	
	F1	L-J	F1	L-J	F1	L-J	F1	L-J	F1	L-J
Qwen3-30B-A3B										
A-Mem	51.50	70.63	35.89	60.87	47.67	55.38	35.45	60.00	46.21	64.85
LiCoMemory	39.88	67.50	31.00	62.32	32.49	44.62	39.84	60.00	38.38	61.15
Mem0	44.39	57.50	38.94	60.87	52.55	50.77	30.50	45.00	44.00	56.05
MemOS	42.14	65.00	34.26	66.67	45.15	46.15	43.97	**75.00**	41.15	62.10
MemoryOS	43.82	63.75	34.76	60.87	41.70	38.46	36.59	60.00	40.93	57.84
MemSkill	41.93	59.37	26.17	55.07	29.60	41.53	36.26	60.00	35.49	54.78
Ours	**56.27**	**80.00**	**40.14**	**73.91**	**53.20**	**69.23**	**51.58**	70.00	**51.79**	**76.79**
LLaMA3.3-70B-Instruct										
A-Mem	46.66	63.12	32.89	55.07	35.81	41.54	24.74	35.00	39.96	55.10
LiCoMemory	40.41	71.88	32.53	65.22	30.69	44.62	**40.22**	**70.00**	35.66	64.85
Mem0	34.04	46.88	29.65	52.17	34.38	29.23	26.05	40.00	32.53	43.95
MemOS	41.89	68.87	36.51	56.52	45.78	41.54	34.34	45.00	41.26	57.96
MemoryOS	45.72	81.25	**38.45**	83.77	39.08	38.92	38.01	65.00	41.92	57.01
MemSkill	43.89	61.87	32.85	66.68	34.99	43.07	21.16	45.00	38.03	57.98
Ours	**52.28**	**89.37**	36.77	**83.78**	**62.88**	**70.76**	40.21	45.00	**50.29**	**66.88**

system maintains a final context set S and scans candidates e_i in descending order of relevance, subject to the constraint

$$\max_{e_j \in S} \cos(v_i, v_j) \leq \delta, \tag{8}$$

where δ is a predefined diversity threshold. If the similarity between e_i and any element in S exceeds δ, e_i is discarded as redundant. This heuristic limits pairwise embedding similarity in the assembled context; its effect on context size and answer quality is evaluated empirically.

4 Experiments

4.1 Experimental setup

Datasets and Baselines. We evaluate 3D-Mem on LoCoMo Maharana et al. (2024), which contains 10 long-horizon conversations, and LongMemEval-S Wu et al. (2024), which contains 500 evaluation examples. Baselines include A-Mem Xu et al. (2025), LiCoMemory Huang et al. (2025), Mem0 Chhikara et al. (2025), MemOS Li et al. (2025), MemoryOS Kang et al. (2025), and MemSkill Zhang et al. (2026).

Implementation Details. We use Qwen3-30B-A3BTeam (2025) and LLaMA3.3-70B-Instruct as LLM backbones, and Qwen3-Embedding-0.6BZhang et al. (2025) for embeddings. Baselines retain recommended method-specific configurations; MemSkill uses a separately trained policy for each backbone. We follow its LoCoMo split (1–6 train, 7–8 validation, 9–10 test). The 3D-Mem controller and skill library are trained with Qwen on LoCoMo and frozen for both backbones and datasets. Retrieval budgets for the graph and L_1–L_5 are $4, 16, 8, 4, 2, 1$ on LoCoMo and $4, 20, 8, 4, 2, 1$ on LongMemEval-S. Generation temperatures are 0.0 for construction and 0.2 for QA. We report F1, BLEU-1, and LLM-as-a-Judge (L-J); training, transfer, and evaluation details appear in Appendix E.

4.2 Main Results

Overall Performance and Generalization. Tables 2 and 3 show strong performance across both datasets and backbones. On LoCoMo, 3D-Mem leads the evaluated memory baselines in overall F1, exceeding the strongest baseline reported in the table by 5.58 points with Qwen (A-Mem) and 8.37 with LLaMA (MemoryOS); it also achieves the highest reported overall L-J. On LongMemEval-S, it achieves the best overall F1 and L-J with Qwen and the best overall L-J with LLaMA, while remaining close to the best F1. The gains vary by category: 3D-Mem trails the strongest baselines on Single-Session Assistant and Preference in LongMemEval-S. Appendix E.4 discusses prompt and reference-length differences. Across three independent Qwen training runs, overall F1 is 0.5132 ± 0.0092 on

Table 3: Results on LongMemEval-S. F1 and L-J are reported as percentages, where L-J denotes the LLM-as-a-Judge score computed with GPT-4o-mini. The best result is shown in bold and the second-best result is underlined.

Method	Single-Session User		Single-Session Assistant		Single-Session Preference		Knowledge Update		Temporal Reasoning		Multi-Session		Overall	
	F1	L-J	F1	L-J	F1	L-J	F1	L-J	F1	L-J	F1	L-J	F1	L-J
Qwen3-30B-A3B														
A-Mem	69.75	**92.86**	77.99	**98.21**	8.99	56.67	54.91	76.92	38.57	56.39	37.03	57.89	47.71	69.80
LiCoMemory	72.51	90.00	**79.97**	94.64	16.41	66.67	58.07	**80.77**	37.42	66.92	33.16	49.62	47.93	70.80
Mem0 w/o Graph	45.19	57.14	11.09	10.71	6.16	30.00	50.07	57.69	32.54	45.86	26.96	36.84	31.57	42.00
MemOS	65.54	88.57	45.79	62.50	12.17	50.00	46.84	71.79	29.30	50.38	36.59	58.65	39.87	62.60
MemoryOS	60.85	74.29	61.02	71.43	7.90	20.00	50.47	69.23	20.57	33.83	35.20	33.83	38.54	50.80
MemSkill	29.94	62.85	24.60	57.14	**17.07**	**80.00**	26.73	50.00	15.50	30.82	15.84	29.32	20.48	43.80
Ours	**76.30**	91.43	42.98	53.57	5.88	50.00	**60.93**	79.49	**52.53**	**78.20**	**48.41**	**70.68**	**52.20**	**73.80**
LLaMA3.3-70B-Instruct														
A-Mem	**72.46**	87.14	**74.43**	**94.64**	4.33	26.67	**61.19**	**75.64**	33.32	36.09	36.54	48.12	**46.86**	58.60
LiCoMemory	62.99	84.29	63.05	85.71	**22.67**	**83.33**	53.91	71.79	38.93	48.12	19.23	48.12	41.12	63.20
Mem0 w/o Graph	68.12	81.43	19.83	17.86	4.95	26.67	56.53	69.23	35.19	44.36	28.09	37.59	37.71	47.60
MemOS	69.13	**88.57**	50.65	62.50	5.91	50.00	57.03	70.51	37.72	50.38	38.25	58.65	44.81	62.40
MemoryOS	48.61	64.29	66.71	80.36	5.33	6.67	37.29	46.15	11.71	15.04	18.14	21.80	28.35	35.40
MemSkill	63.81	84.28	45.72	75.00	21.39	80.00	39.17	64.10	28.28	57.89	10.28	44.36	31.70	62.20
Ours	72.32	84.28	34.49	48.21	4.10	43.33	59.09	**75.64**	**43.12**	**59.39**	**42.26**	**60.15**	46.16	**63.40**

Table 4: Deployment-stage cost and answer quality on LoCoMo with Qwen3-30B-A3B, excluding one-time training cost. † marks methods with task-specific policy training. Best results are boldfaced and second-best results underlined.

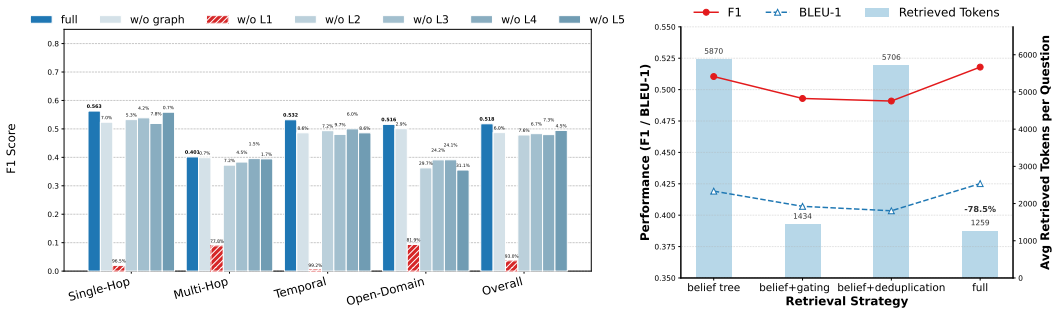
Baseline	Avg Build Time (s) ↓	Avg Build Tokens ↓	Avg QA Retrieved Tokens Mean ↓	F1 ↑	BLEU-1 ↑
A-Mem	2480.95	1,243,205	1851.47	0.4619	0.3858
LiCoMemory	**501.65**	889,266	2039.27	0.3639	0.3013
Mem0	1211.15	380,524	1661.11	0.4402	0.3710
MemoryOS	9566.79	2,819,435	6807.81	0.4093	0.3394
MemSkill†	647.80	**249,810.5**	**605.60**	0.3549	0.3012
Ours†	611.99	359,421.0	1259.25	**0.5179**	**0.4251**

LoCoMo and 0.5098 ± 0.0107 on LongMemEval-S (Appendix G.3). Full-history comparisons appear in Appendix G.1.

Deployment-Stage Cost. Table 4 compares deployment-stage memory construction, retrieved context, and answer quality under matched serving settings, excluding one-time training cost. 3D-Mem achieves the highest F1 and BLEU-1 among the listed methods. Relative to MemSkill, it improves F1 by 0.1630 and BLEU-1 by 0.1239 with 5.5% lower memory-build time, while using more construction and retrieval tokens. This comparison characterizes a quality–cost trade-off between complete frameworks; the ablations below examine individual components.

4.3 Ablation Study

Decision Mechanisms and Graph-to-Source Projection. Table 5 reports three-run ablations. The Retrieval ablation retains the same constructed memory, including graph contents and source anchors, and disables only the graph-to-source relevance term $\Delta_{\mathcal{G}}(c)$ in Eq. 7; the remaining retrieval and assembly procedures are unchanged. F1 decreases from 0.513 to 0.503 despite retaining the same stored information. This supports the graph's role in guiding source-evidence selection through provenance projection, complementing the graph-removal ablation in Figure 2a. Removing Evolution lowers F1 by 0.025. Without Assembly, F1 decreases by 0.017 and retrieved context increases 4.45-fold. Together, these results support adapting memory construction, using graph relevance for evidence access, and filtering the resulting context.



(a) Memory construction ablation.

(b) Question-answering retrieval ablation.

Figure 2: Ablation results for memory construction and question-answering retrieval.

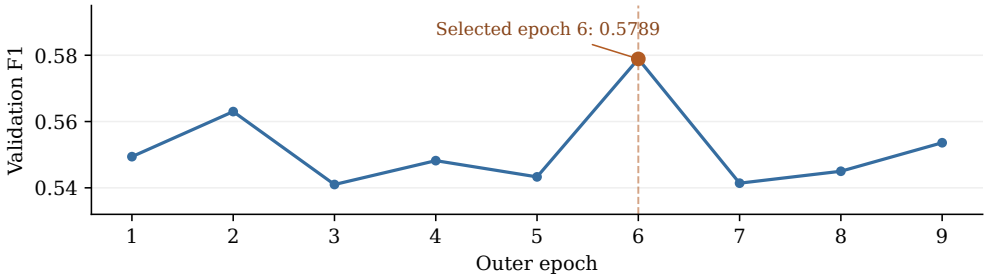


Figure 3: Validation F1 across nine outer epochs. Epoch 6 attains the highest value and is selected for evaluation; later operator updates do not improve validation F1.

Operator-Library Evolution. With graph architecture, retrieval configuration, training data, and PPO budget unchanged, fixing the initial operator library lowers F1 by 0.0111 (1.11 percentage points). This controlled comparison supports the incremental value of operator induction; repeated-run uncertainty for this particular contrast is not available. A separate fixed-library selector comparison in Appendix F evaluates the PPO controller against an LLM-as-policy selector.

Memory Construction. Figure 2a examines the contribution of each memory component. Removing L_1 reduces overall F1 by 93.0%, highlighting its role as the primary source of fine-grained answer evidence. Removing the graph reduces overall F1 by 6.0%, consistent with its role in organizing and locating evidence. Higher temporal levels support broader context and cross-temporal aggregation: removing L_2 or L_5 reduces open-domain F1 by 29.7% or 31.1%, respectively.

Question-Answering Retrieval. As shown in Figure 2b, the full retrieval strategy achieves the best quality–efficiency trade-off. It achieves the best answer quality while retrieving only 1,259 tokens per question on average. Compared with belief-only retrieval, it reduces the retrieved context by 78.5%. Adding gating alone reduces the average retrieval budget to 1,434 tokens, indicating that it effectively controls retrieval cost. Building on this, the full strategy further reduces the token count to 1,259 through information-redundancy-based deduplication while preserving the best answer quality.

4.4 Dynamics of Memory Skill Evolution

Figure 3 shows the validation F1 trajectory for the nine outer epochs. Epoch 6 is selected solely by validation F1, with the highest value of 0.5789 (Appendix G.2). The library at this epoch includes specialized extraction operators; the trajectory does not establish that increasing skill complexity improves performance.

Appendix D illustrates changes in evidence presentation: Epoch 5 already retrieves `Ferrari`, but the generator produces a generic answer; Epoch 6 places the name in an explicit purchase-event

Table 5: Ablations of decision mechanisms (mean ± standard deviation over three independent runs). The w/o Graph Projection variant disables only graph relevance projected to source chunks, retaining the same constructed memory.

Setting	F1 ↑	BLEU-1 ↑	Retrieved tokens/QA ↓
3D-Mem	$\mathbf{0.513} \pm 0.009$	$\mathbf{0.423} \pm 0.005$	$\mathbf{1{,}282.3} \pm 34.7$
w/o Evolution Decision	0.488 ± 0.010	0.400 ± 0.011	$1{,}400.9 \pm 129.9$
w/o Graph Projection	0.503 ± 0.005	0.400 ± 0.010	$1{,}372.7 \pm 61.0$
w/o Assembly Decision	0.496 ± 0.005	0.406 ± 0.004	$5{,}712.2 \pm 5.7$

description and yields the specific answer. This example involves both memory representation and answer generation.

5 Conclusion

We presented **3D-Mem**, which learns to organize conversational evidence for future retrieval. Task feedback guides graph-operator induction, and provenance links carry the resulting graph relevance into source-evidence selection. Hierarchical retrieval and context assembly make this evolving index usable for answering. Evaluations on LoCoMo and LongMemEval-S support the framework's answer quality and compact contexts; controlled library ablation supports the incremental value of adapting the operations that organize evidence.

Limitations. Retrieval scoring and assembly prompts are fixed, and the decisions are not jointly optimized. Evaluation covers two benchmarks and two backbones; LoCoMo has only two test conversations, with correlated questions. Baseline prompts differ, and the reported runs do not establish statistical significance for all comparisons. The evaluated baselines do not include MAGMA, Memory-R1, or Mem-α. Deployment costs exclude policy training and operator induction. These limits constrain conclusions about generalization, comparative superiority, and total computational cost.

AI Use Statement

We used generative AI tools to assist with translation and language polishing of the manuscript.

References

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.

Zijie Dai et al. Recmem: Recurrence-based memory consolidation for efficient and effective long-running LLM agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026. URL https://aclanthology.org/2026.findings-acl.1619/.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018. URL https://arxiv.org/abs/1802.06070.

Jiale Han, Austin Cheung, Yubai Wei, Zheng Yu, Xusheng Wang, Bing Zhu, and Yi Yang. RAG meets temporal graphs: Time-sensitive modeling and retrieval for evolving knowledge. *arXiv preprint arXiv:2510.13590*, 2025. URL https://arxiv.org/abs/2510.13590.

Xanh Ho, Jiahao Huang, Florian Boudin, and Akiko Aizawa. LLM-as-a-judge: Reassessing the performance of LLMs in extractive QA. *arXiv preprint arXiv:2504.11972*, 2025. URL https://arxiv.org/abs/2504.11972.

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.

Zhengjun Huang, Zhoujin Tian, Qintian Guo, Fangyuan Zhang, Yingli Zhou, Di Jiang, Zeying Xie, and Xiaofang Zhou. Licomemory: Lightweight and cognitive agentic memory for efficient long-term reasoning. *arXiv preprint arXiv:2511.01448*, 2025.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. MAGMA: A multi-graph based agentic memory architecture for AI agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36848–36865. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1709. URL https://aclanthology.org/2026.acl-long.1709/.

Tobias Jung, Daniel Polani, and Peter Stone. Empowerment for continuous agent–environment systems. *Adaptive Behavior*, 19(1):16–39, 2011. doi: 10.1177/1059712310392389. URL https://doi.org/10.1177/1059712310392389.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent, 2025. URL https://arxiv.org/abs/2506.06326.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474, 2020.

Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, Chengbao Liu, and Jie Tan. Timem: Temporal-hierarchical memory consolidation for long-horizon conversational agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 21700–21720. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.1091. URL https://aclanthology.org/2026.findings-acl.1091/.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, et al. Memos: An operating system for memory-augmented generation (mag) in large language models. *arXiv preprint arXiv:2505.22101*, 2025.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Wenquan Ma, Jiayan Nan, and Wenlong Wu. What deserves memory: Adaptive memory distillation for LLM agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 34789–34812. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1607. URL https://aclanthology.org/2026.acl-long.1607/.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. *arXiv preprint arXiv:2402.17753*, 2024.

James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. Memgpt: towards llms as operating systems. 2023.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 188–204. PMLR, 2021. URL https://proceedings.mlr.press/v155/pertsch21a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019. URL https://arxiv.org/abs/1907.01657.

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

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025. URL https://arxiv.org/abs/2509.25911.

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

Zhaofen Wu, Hanrong Zhang, Fulin Lin, Wujiang Xu, Xinran Xu, Yankai Chen, Henry Peng Zou, Shaowen Chen, Weizhi Zhang, Xue Liu, et al. Gam: Hierarchical graph-based agentic memory for llm agents. *arXiv preprint arXiv:2604.12285*, 2026.

Haidong Xin, Xinze Li, Zhenghao Liu, Yukun Yan, Shuo Wang, Cheng Yang, Yu Gu, Ge Yu, and Maosong Sun. Metamem: Evolving meta-memory for knowledge utilization through self-reflective symbolic optimization. *arXiv preprint arXiv:2602.11182*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025. URL https://arxiv.org/abs/2508.19828.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. Memskill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

APPENDIX

A RELATED WORK

Memory Architectures for Autonomous Agents. To enable long-horizon capabilities in LLM-based agents, external memory modules have become essential components. Early vector-based systems, such as standard RAG and MemGPT (Lewis et al., 2020; Packer et al., 2023), provide mechanisms for storing and accessing external context. Recent systems add temporal, hierarchical, or graph structure: TiMem and TG-RAG model temporal organization, MemOS provides a hierarchical store, A-Mem evolves graph links and attributes, and MAGMA retrieves over multiple predefined graph views (Li et al., 2026; Han et al., 2025; Li et al., 2025; Xu et al., 2025; Jiang et al., 2026). 3D-Mem extends this line of work by using downstream feedback to evolve the provenance-preserving graph operations available to its selector. Applying these operations changes the relations and source anchors used to score evidence during retrieval.

Memory adaptation has likewise been studied directly. Memory-R1 learns structured memory management, Mem-α learns memory-construction decisions, and NEMORI distills memories based on future predictive value (Yan et al., 2025; Wang et al., 2025; Ma et al., 2026). MemSkill combines context-dependent skill selection with a frozen LLM designer that refines and extends a skill bank from difficult cases (Zhang et al., 2026). 3D-Mem adopts this controller–designer pattern for explicit graph operators with target views, readable sources, permitted edits, and provenance requirements. The resulting topology is used as a semantic evidence index: graph relevance is projected to supporting L_1 chunks, combined with hierarchical relevance, and filtered for cross-view redundancy. This connection between operator induction, graph structure, and evidence access defines the framework's contribution.

Skill-Based Learning and Operator Induction. Skill-based RL learns reusable behaviors and priors that guide downstream control (Pertsch et al., 2021). Unsupervised approaches emphasize diverse or predictable behaviors (Eysenbach et al., 2018; Sharma et al., 2019), while empowerment measures an agent's observable influence on its environment (Jung et al., 2011). 3D-Mem considers explicit memory-editing operators guided by QA failures. Their usefulness depends on the evidence structures they create and the answers these structures support. The evolving library specifies execution guidance over graph-edit primitives, with source anchors connecting structural changes to retrievable evidence.

Retrieval Granularity and Addressing Mechanisms. Identifying relevant information from massive historical contexts remains challenging. Standard dense retrieval relies on fixed-size chunks and top-k similarity thresholds (Hu et al., 2025); other systems add metacognitive routing, temporal graphs, or graph-guided retrieval (Xin et al., 2026; Han et al., 2025; Jiang et al., 2026; Wu et al., 2026). 3D-Mem combines evidence-count-calibrated branch scores with graph relevance projected through provenance anchors. Additive smoothing moderates low-evidence branches, while provenance support reconnects structural relevance to source evidence. Together, these signals guide hierarchical retrieval over the topology produced by the evolving graph-operation skills.

Context Management and Redundancy Elimination. Long contexts can make relevant evidence harder to use (Liu et al., 2024). Context-management approaches include compression, sliding windows, virtual context paging, and similarity-based filtering (Packer et al., 2023). 3D-Mem combines a prompted utility gate with a greedy embedding-similarity filter to limit irrelevant or repeated evidence across memory views. These mechanisms are heuristics; the Assembly ablation measures their aggregate effect on answer quality and context size.

B 3D-MEM SYSTEM PROMPTS

We present key prompt templates used in 3D-Mem's hierarchical consolidation pipeline. Each prompt implements the layer-specific instruction I_k in the consolidation formula $L_k = \Phi_k(C_k, H_k, I_k)$.

B.1 L1 Segment Memory Consolidator

L1 Segment Consolidator

You are a dialogue memory generator. Your task is to write a fragment memory that captures only the NEW facts from the "Current Conversation" (do not repeat anything already covered in "Historical Memories").
Core principle: Convert dialogue from first-person to third-person narration, preserving as much substantive information content from the original as possible, excluding only confirmed non-informative words.
What to preserve:

- All substantive information: people, events, times, places, causes, results, numbers, specific descriptions
- Original wording: Keep specific terms used in dialogue for titles, item names, activity descriptions, etc. Numbers use Arabic numerals
- Emotional expressions: Retain explicit emotions and attitudes from original (like "happy", "worried", "likes"), but avoid adding subjective inferences not present in original

What to exclude: Only exclude purely functional words: greetings ("hi", "bye"), confirmation words ("uh-huh", "okay", "yes"), meaningless fillers ("um", "you know", "like")
Time normalization: Preserve the original relative time expressions exactly as written (e.g., "last night", "this morning", "last Friday"). DO NOT convert relative time to absolute dates.
Style:

- Use English third-person narration.
- Write plain sentences (no lists/numbering/Markdown). Aim for 2-4 sentences, but allow longer to retain essential details.
- Use exact proper nouns as in the dialogue; do not replace/expand/infer names, organizations, or locations.
- Each memory should focus on one core fact or closely related fact group. avoid packing too many unrelated details into a single entry.

Inputs:

- Historical memories (do not repeat): `{history}`
- Current conversation: `{dialog}`

Please generate a fragment memory that contains ONLY the new facts. If the current conversation has no substantial new content, provide a minimal 1-2 sentence summary of the core topic or attitude expressed in this turn (do NOT output "no significant additions" or similar empty statements).

B.2 L2 Session Summary Consolidator

L2 Session Consolidator

You are a session memory consolidator. Your task is to merge multiple segment memories from a single conversation session into a coherent, non-redundant event summary.
Instructions:

1. Combine related facts from the segment memories below
2. Remove redundant information while preserving all unique facts
3. Maintain chronological order of events within the session
4. Use third-person narration
5. Create a cohesive narrative that captures the key events and topics discussed

Inputs:

- Historical context (for continuity, do not repeat): `{history}`
- Segment memories to consolidate: `{children}`

Generate a consolidated session summary (3-6 sentences).

B.3 L3 Daily Pattern Consolidator

L3 Daily Pattern Consolidator

You are a daily pattern analyzer. Your task is to analyze multiple session memories from a single day and identify daily patterns, routines, and recurring interests.

Instructions:

1. Identify any routine activities or regular patterns
2. Note recurring topics, interests, or concerns
3. Capture the user's mood or emotional state for the day
4. Highlight any significant events or decisions
5. Use third-person narration

Inputs:

- Historical context (for continuity): `{history}`
- Session memories from this day: `{children}`

Generate a daily summary capturing patterns and key events (3-5 sentences).

B.4 L4 Weekly Behavioral Consolidator

L4 Weekly Behavioral Consolidator

You are a weekly behavior analyst. Your task is to analyze daily memories from a week and identify evolving behavioral features and preference patterns.

Instructions:

1. Identify consistent behavioral patterns across the week
2. Note any evolving preferences or interests
3. Capture recurring themes or concerns
4. Highlight any significant changes or developments
5. Focus on patterns rather than individual events
6. Use third-person narration

Inputs:

- Historical context (for continuity): `{history}`
- Daily memories from this week: `{children}`

Generate a weekly behavioral summary capturing patterns and trends (4-6 sentences).

B.5 L5 Profile Memory Consolidator

L5 Profile Consolidator

You are a persona profiler. Your task is to create or update a comprehensive user profile based on long-term behavioral patterns and preferences.

Create a structured profile with the following sections:

1. Basic Identity

- Role/Occupation (if known)
- Life context (living situation, family, etc.)
- Key social contacts

2. Personality Traits

- Communication style
- Decision-making patterns
- Emotional tendencies

3. Interests & Preferences

- Hobbies and activities
- Topics of interest
- Preferences (food, entertainment, etc.)

4. Values & Goals

- Core values
- Short-term goals
- Long-term aspirations

5. Key Life Events (this period)
- Significant events with dates
- Important decisions or changes

Inputs:
- Previous profile (update incrementally): `{history}`
- Weekly behavioral summaries: `{children}`

Generate an updated user profile.

C Initial Action Space for Graph Evolution

The initial action space $\mathcal{A}_0$ consists of six graph management skills, each with associated arguments and grounded instruction templates. Table 6 provides the formal definitions.

Table 6: Initial Action Space $\mathcal{A}_0$ for Policy-Driven Graph Evolution

Skill	Signature	Description
`add_entity`	(n, τ, c)	Insert entity node with name n, type τ, and content c
`add_relation`	(e_s, e_t, τ, c)	Insert typed edge from source e_s to target e_t
`merge_entity`	$(e_{\text{src}}, e_{\text{tgt}})$	Consolidate duplicate entity e_{src} into canonical e_{tgt}
`update_relation`	(r, δ)	Revise relation r with enriched content δ
`delete_graph_object`	(o)	Remove outdated or contradicted graph object o
`noop`	$()$	Confirm no graph modification needed

Each skill is associated with: (i) a *target view* indicating which memory axis to modify (graph or retrieval), (ii) a set of *readable views* specifying accessible context (chunk, summary, graph), and (iii) an *instruction template* that guides the LLM executor to generate valid arguments. The full instruction templates are provided below.

Skill Instruction Templates.

- `add_entity`: Capture a distinct entity (person, organization, place, concept) from the current chunk as a grounded graph node. Output fields: `ENTITY_NAME`, `ENTITY_TYPE`, `CONTENT`, `SOURCE_CHUNK_IDS`, `CONFIDENCE`.
- `add_relation`: Capture an explicit relationship between two entities as a grounded graph edge. Reuse retrieved entity names when referring to the same entities. Output fields: `SOURCE_ENTITY`, `TARGET_ENTITY`, `RELATION_TYPE`, `CONTENT`, `SOURCE_CHUNK_IDS`, `CONFIDENCE`.
- `merge_entity`: Consolidate duplicate graph entities (aliases, spelling variants) that refer to the same real-world object. Output fields: `SOURCE_ENTITY`, `TARGET_ENTITY`, `CONTENT`.
- `update_relation`: Revise a retrieved graph relation when the current chunk corrects, enriches, or clarifies it. The `RELATION_ID` must be copied exactly from retrieved references. Output fields: `RELATION_ID`, `RELATION_TYPE`, `CONTENT`, `SOURCE_CHUNK_IDS`, `CONFIDENCE`.
- `delete`: Remove a graph entity or relation only when current evidence shows it is invalid or contradicted. The `OBJECT_ID` must come from retrieved graph memory. Output fields: `OBJECT_ID`.
- `noop`: Confirm that no graph update is needed—either the current chunk contains no durable information, or retrieved memory already captures the relevant facts. Output fields: none.
- `extract_event_entity`: Extract a concrete event or personal milestone from the current chunk as a grounded graph node. This skill is used when the dialogue describes a dated or contextualized experience, such as a trip, appointment, health event, purchase, hobby activity, or major life update. Output fields: `EVENT_NAME`, `EVENT_TYPE`, `CONTENT`, `SOURCE_CHUNK_IDS`, `CONFIDENCE`.

Table 7: Skill Evolution Trace Across Training Epochs

Skill	First Appeared Epoch	Updated Epoch(s)	Deleted Epoch
add_entity	Initial	–	–
add_relation	Initial	Epoch 1, 3, 8, 9	–
merge_entity	Initial	–	–
update_relation	Initial	–	–
delete_graph_object	Initial	–	–
noop	Initial	–	–
extract_event_entity	Epoch 1	Epoch 2, 4, 7	–
model_activity_sequence	Epoch 2	Epoch 3, 8, 9	–
extract_book_entity	Epoch 4	–	Epoch 5
extract_numeric_entity	Epoch 5	–	–
extract_allergy_specific_entity	Epoch 6	–	Epoch 7
extract_pet_adoption_challenge	Epoch 6	–	Epoch 7

- model_activity_sequence: Represent a sequence of related activities or behavioral changes as structured graph memory. This skill links multiple steps or recurring actions across time, such as moving from a health check-up to exercise planning, or from a hobby discussion to an actual activity. Output fields: SEQUENCE_NAME, SOURCE_EVENTS, TARGET_EVENTS, CONTENT, SOURCE_CHUNK_IDS, CONFIDENCE.
- extract_book_entity: Extract book-related information as graph entities when the dialogue contains book titles, reading interests, recommendations, or discussion of books. This skill captures literary objects and reading preferences that may otherwise be treated as generic concepts. Output fields: BOOK_ENTITY, BOOK_TYPE, CONTENT, SOURCE_CHUNK_IDS, CONFIDENCE.
- extract_numeric_entity: Extract explicit numerical information and normalized values from the current chunk as graph entities. This skill is used for dates, counts, quantities, durations, scores, or other numeric attributes that should be preserved for precise retrieval and question answering. Output fields: NUMERIC_ENTITY, VALUE, UNIT, CONTENT, SOURCE_CHUNK_IDS, CONFIDENCE.
- extract_allergy_specific_entity: Extract allergy-related facts as specialized graph entities. This skill captures allergens, symptoms, dietary or medical restrictions, and relevant health constraints when they are explicitly mentioned in the dialogue. Output fields: ALLERGY_ENTITY, ALLERGY_TYPE, CONTENT, SOURCE_CHUNK_IDS, CONFIDENCE.
- extract_pet_adoption_challenge: Extract pet-adoption-related situations as graph entities, especially when the dialogue describes adoption plans, obstacles, preferences, responsibilities, or constraints. This skill preserves adoption challenges that may be important for later personal-memory questions. Output fields: ADOPTION_ENTITY, CHALLENGE_TYPE, CONTENT, SOURCE_CHUNK_IDS, CONFIDENCE.

Skill Evolution Process. Beyond the initial action space, the skill library is dynamically maintained during training. In addition to generating new skills, the Meta-Reflector can refine existing skills when they remain useful but require more specific instructions, and remove low-utility or overly specialized skills in later epochs. Table 7 summarizes initialization, refinement, generation, and deletion across the nine training epochs.

Skill Encoding. Each instruction template is encoded into a dense vector $\psi_{\text{skill}}(a_i) \in \mathbb{R}^d$ using the same pre-trained language model encoder. The dual-encoder policy computes selection logits as $z_{t,i} = \text{MLP}_{\text{inter}}(s_t \oplus \psi_{\text{skill}}(a_i))$, enabling dynamic adaptation when the action space evolves during meta-learning.

D Case Study on Evidence Presentation and Answer Specificity

We present a qualitative example of memory representation and answer specificity across checkpoints. The question asks what Calvin recently obtained that he described as a "masterpiece on wheels"; the reference answer is Ferrari. The excerpts show the specific entity appearing in

retrieved memories before the generator produces that answer, allowing us to distinguish evidence presence from successful use.

Case Study: Memory Representation and Answer Specificity

Note: The full retrieval list contains many memories; for readability, we only show the most relevant retrieved memories for each epoch.
Question: What did Calvin recently get that is a "masterpiece on wheels"?
Ground Truth: Ferrari

epoch 1: skill `add_entity`, `add_relation`, `update_relation`
→ retrieved memory:
rank 1: `Calvin's Tokyo Performance`, content: Calvin's upcoming performance in Tokyo this month.
rank 2: `Red Sports Car`, content: A car Calvin owns and showed to Dave.
rank 20: edge `Calvin -> Red Sports Car`, content: Calvin owns the Red Sports Car.
Answer: Red Sports Car (wrong)

epoch 2: skill `extract_event_entity`, `add_entity`, `add_relation`
→ retrieved memory:
rank 1: `Orange Car`, content: A shiny car Calvin shared in a photo.
rank 5: `Calvin's Red Sports Car`, content: A red sports car Calvin is proud of and associates with hard work.
rank 10: `chunk_D13_c005`, content: Calvin said, "customizing a masterpiece with those small details is what makes it unique and personalized."
Answer: red sports car (wrong)

epoch 3: skill `model_activity_sequence`, `extract_event_entity`, `add_relation`
→ retrieved memory:
rank 2: `car_event_2023-03-23`, content: Calvin attended a car lover's event with classic cars.
rank 4: `ferrari_2023`, content: Calvin's new black sports car purchased recently.
rank 10: `chunk_D13_c005`, content: Calvin discussed customizing a masterpiece with small details.
Answer: Calvin's new black sports car (wrong)

epoch 4: skill `extract_event_entity`, `model_activity_sequence`, `add_relation`
→ retrieved memory:
rank 1: `diamond pendant necklace`, content: Calvin's gold necklace with diamond pendant.
rank 2: `car repairs`, content: Calvin's car repairs.
rank 5: `Ferrari`, content: Calvin's new black sports car.
Answer: Calvin's new black sports car (wrong)

epoch 5: skill `extract_event_entity`, `model_activity_sequence`, `extract_numeric_entity`
→ retrieved memory:
rank 1: `Ferrari`, content: Calvin's new sports car, described as a "masterpiece on wheels".
rank 4: `Calvin_Red_Ferrari`, content: A red Ferrari owned by Calvin, shared in a photo during a garage visit.
rank 10: `chunk_D13_c005`, content: Calvin discussed customizing a masterpiece with small details.
Answer: new sports car (wrong)

epoch 6: skill `extract_event_entity`, `extract_numeric_entity`, `model_activity_sequence`, `add_relation`
→ retrieved memory:
rank 1: `Calvin_Ferrari_Purchase_2023`, content: Calvin purchased a new Ferrari, describing it as a masterpiece on wheels.
rank 5: `Calvin's_New_Car_Acquisition`, content: Calvin acquired a luxury sports car, described as "amazing", "smooth", and "powerful".
rank 24: edge `Calvin_Music_Career_2023 -> Calvin_Ferrari_Purchase_2023`, content: Calvin's music success coincided with personal achievements like the Ferrari purchase.
Answer: Ferrari (right)

In Epochs 1–2, the displayed retrieved memories use broad vehicle descriptions, and the answers remain generic. Epochs 3–5 already retrieve Ferrari-related entries. In particular, Epoch 5 ranks a node named `Ferrari` first and includes the phrase "masterpiece on wheels", yet the answer is "new sports car". Thus, this checkpoint cannot be characterized simply as a failure to retrieve the entity.

At Epoch 6, the top event-level entry explicitly states that Calvin purchased a Ferrari and connects this event to the descriptive phrase. The generated answer then matches the reference. This is consistent with evidence presentation affecting answer specificity, but the example does not isolate retrieval, wording, and generation effects. It is illustrative rather than a causal or statistical test. The fixed-library and graph-projection ablations in Section 4.3 provide controlled aggregate comparisons for operator evolution and evidence access, respectively.

E Implementation Details

E.1 Training and Transfer Protocol

The controller and graph-operation skill library are trained with Qwen3-30B-A3B on LoCoMo conversations 1–6, validated on 7–8, and tested on 9–10. Both components remain frozen for evaluation with Qwen3-30B-A3B and LLaMA3.3-70B-Instruct on LoCoMo and LongMemEval-S. LLaMA evaluates cross-backbone transfer; LongMemEval-S additionally evaluates cross-dataset transfer. For MemSkill, we train separate controllers with each backbone following its recommended configuration.

E.2 Training Setup

We train the dual-encoder policy on the LoCoMo training split with Qwen3-30B-A3B as the LLM backend and Qwen3-Embedding-0.6B for state and operation encoding. Training uses 9 outer epochs, each containing 25 inner epochs; the reported checkpoint is outer epoch 6. The default random seed is 42. Table 5 and Appendix G.3 separately report three-run statistics. Table 8 summarizes the RL procedure, and Table 9 gives its hyperparameters. We use Adam and two PPO update epochs, with GAE, clipped PPO, entropy regularization, and gradient clipping. For training and evaluation, memory retrieval considers 20 candidates.

Table 8: RL training procedure for the Evolution policy.

Component	Specification
Episode / step	One complete LoCoMo conversation / one session.
State	Concatenated embeddings of the current session and retrieved chunk, summary, and graph views.
Action	Gumbel-Top-k selection of three skills, followed by graph-action generation with the LLM executor.
Rollouts	16×25 conversation rollouts per outer epoch; 9 outer epochs.
Reward	Step-level skill–executor consistency shaping and conversation-level QA F1.
Credit assignment	60% of delayed QA reward is assigned to the final step; the remaining 40% is distributed across steps with decay factor 0.9.
Optimization	GAE with clipped PPO.

E.3 Retrieval Scoring for L_1 Chunks

During retrieval, we re-rank L_1 chunk candidates by combining (i) a semantic text score, (ii) structural support from top-matched relations, and (iii) the additively smoothed score of the parent branch. Raw cosine similarity $r \in [-1, 1]$ is mapped as $s = (r + 1)/2$ without clipping. Concretely, for a chunk c, we compute:

$$s_{\text{final}}(c) = w_{\text{text}} \cdot s_{L_1}(c) + (1 - w_{\text{text}}) \cdot \bar{s}_{\text{parent}(c)}, \tag{9}$$

$$s_{L_1}(c) = w_{\text{text}} \cdot s_{\text{text}}(c) + (1 - w_{\text{text}}) \cdot s_{\text{struct}}(c), \tag{10}$$

$$s_{\text{struct}}(c) = \text{mean}\big(\text{top-}M \text{ relation supports of } c\big), \tag{11}$$

We set $w_{\text{text}} = 0.7$ and $M = 2$. Expanding the nested scores gives Eq. 7 with $\omega_1 = 0.49$, $\omega_2 = 0.30$, and $w_g = 0.21$. We use a minimum L_1 score threshold of 0.3 (i.e., `mts_l1_threshold`=0.3) and enforce L_1 as the primary evidence level (`mts_force_l1`=True). The remaining implementation parameters are `mts_prior_alpha`=`mts_prior_beta`=1.0, `mts_min_evidence`=1, and `mts_exploration_bonus`=1.5.

Table 9: Training configuration for the 3D-Mem dual-encoder policy model.

Configuration	Value
Dataset	LoCoMo
Training phase	`memory_management`
Session mode	Full-session
LLM backend	Qwen3-30B-A3B
State / operation encoder	Qwen3-Embedding-0.6B
Outer epochs	9
Evaluated outer epoch	6
Inner epochs per outer epoch	25
Learning rate	1×10^{-4}
Batch size / minibatch size	16 / 32
PPO update epochs	2
Discount factor γ	0.99
GAE λ	0.95
PPO clip ϵ	0.2
Entropy / value coefficient	0.01 / 0.5
Target KL / max grad norm	0.02 / 0.5
Task reward metric	F1
Default random seed	42
Action top-k during training	3
Chunk size / overlap	2048 / 256
Memory top-k / eval top-k	20 / 20
Write top-k chunks / summaries / graph	5 / 3 / 8
Query top-k chunks / summaries / graph	8 / 3 / 8
Meta-Reflector frequency	Every outer epoch
Meta-Reflector reflection cycles / max changes	3 / 2

Pseudo-count formulation. The branch score in Eq. 6 also admits a Beta$(1, 1)$ fractional-pseudo-count representation. Defining $\alpha_m = 1 + \sum_{c \in \mathrm{Desc}(m)} s_c$ and $\beta_m = 1 + \sum_{c \in \mathrm{Desc}(m)} (1 - s_c)$ yields $\bar{s}_m = \alpha_m / (\alpha_m + \beta_m)$. The two unit pseudo-counts center an unsupported branch at 0.5, while accumulated similarity evidence determines the strength of its deviation from this neutral value.

E.4 Category Sizes and Baseline Prompts

LongMemEval-S contains 500 examples: 133 Temporal Reasoning, 133 Multi-Session, 78 Knowledge Update, 70 Single-Session User, 56 Single-Session Assistant, and 30 Single-Session Preference. The two smallest categories are more sensitive to individual prediction changes.

We retain official baseline prompts, whose answer-length constraints differ: MemSkill has no explicit limit, whereas A-Mem does. Our QA prompt asks for answers under 5–6 words. Single-Session Preference references average 53.9 tokens, so brevity can limit coverage of multi-fact answers and affect lexical metrics. This is a plausible contributor to the weak Preference results, not an experimentally isolated explanation. Method-specific prompt comparisons assess complete pipelines; a common-prompt evaluation would be needed to isolate memory quality from answer-generation instructions.

E.5 Metrics

We report token-level F1, BLEU-1, and LLM-as-a-Judge (L-J), following recent memory studies (Zhang et al., 2026; Ma et al., 2026; Jiang et al., 2026; Dai et al., 2026). F1 and BLEU-1 measure lexical overlap, whereas L-J assesses semantic correctness, including paraphrases and equivalent date formats. These complementary measures have different absolute score scales; related work examines this distinction in extractive QA (Ho et al., 2025). All methods and datasets use the same normalization, matching, judge, and evaluation criteria.

Token Normalization. For F1 and BLEU-1, both the generated answer and the gold answer are first converted into lowercase alphanumeric tokens:

$$T(x) = \text{AlnumTokenize}(\text{Lowercase}(x)).$$

This normalization ignores casing, punctuation, and surface-level formatting differences, so the lexical metrics focus on content-bearing tokens.

F1. Let $\hat{Y}$ and Y denote the token sets of the generated answer and gold answer. Precision and recall are computed as

$$P = \frac{|\hat{Y} \cap Y|}{|\hat{Y}|}, \qquad R = \frac{|\hat{Y} \cap Y|}{|Y|}.$$

The F1 score is

$$\text{F1} = \frac{2PR}{P+R}.$$

If either answer contains no valid token, the score is set to 0. Repeated tokens are not counted multiple times under this F1 protocol.

BLEU-1. BLEU-1 is computed as clipped unigram precision with a brevity penalty. Let $c_{\hat{y}}(w)$ and $c_y(w)$ be the unigram counts of token w in the generated and gold answers. The clipped unigram match count is

$$m = \sum_w \min(c_{\hat{y}}(w), c_y(w)).$$

The unigram precision is

$$p_1 = \frac{m}{|\hat{y}|}.$$

The brevity penalty is

$$\text{BP} = \begin{cases} 1, & |\hat{y}| \geq |y|, \\ \exp(1 - \frac{|y|}{|\hat{y}|}), & |\hat{y}| < |y|. \end{cases}$$

The final BLEU-1 score is

$$\text{BLEU-1} = \text{BP} \cdot p_1.$$

LLM-as-a-Judge. For L-J, we use `GPT-4o-mini` as the judge model. The judge receives the question, the gold answer, and the generated answer, and assigns one of two labels: `CORRECT` or `WRONG`. A generated answer is judged as correct if it refers to the same target information as the gold answer, even when it is longer, phrased differently, or contains additional context. For time-related questions, different surface forms are treated as correct when they refer to the same date, month, year, or time period.

The binary label is mapped to a numerical score:

$$s_i = \begin{cases} 1, & \text{if the label is } \texttt{CORRECT}, \\ 0, & \text{if the label is } \texttt{WRONG}. \end{cases}$$

The final L-J score is the average accuracy over all judged questions:

$$\text{L-J} = \frac{1}{|\mathcal{Q}_{\text{judged}}|} \sum_{i \in \mathcal{Q}_{\text{judged}}} s_i.$$

Judge Prompt. The unified judge prompt is summarized below.

Unified LLM-as-a-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".

E.6 PROMPTS

Meta-Reflector

Part I: Reflection Analysis

Role. You are in reflection cycle `{reflection_round}/{reflection_round_total}` for analyzing failure cases and memory-management skills. Critique the previous analysis and improve it.

Inputs.

- **Previous analysis:** `{analysis_feedback}`
- **Current memory-management skills:** `{operation_bank_description}`
- **Operation-evolution feedback:** `{evolution_feedback}`
- **Failure cases:** `{failure_cases_details}` for `{num_failure_cases}` cases.

Reflection instructions.

- Check for missing or misclassified failure patterns.
- Validate `root_cause` labels against the cases and retrieved memories.
- Strengthen `potential_fix` suggestions so they are specific and actionable.
- Keep the same output format and output only JSON.
- Provide up to `{max_changes}` recommendations total; use fewer if only one change is justified.
- Apply any additional hint in `{new_skill_hint}`.

Required JSON schema.

```
{
"failure_patterns": [
{
"pattern_name": "<descriptive name>",
"affected_cases": [<case ids>],
"root_cause": "<graph_construction_failure |
retrieval_failure | memory_quality_failure>",
"explanation": "<why this failure pattern occurs>",
"potential_fix": "<operation change that could address
it>"
}
],
"recommendations": [
{
"action": "<add_new_operation |
refine_existing_operation | no_change>",
"target_operation": "<operation name or null>",
"rationale": "<why this is the best improvement>",
"priority": "<high | medium | low>"
}
],
"summary": "<1-2 sentence summary>"
}
```

Constraint. Output only the JSON object, with no additional text.

Part II: Operation Refinement

Role. Based on the failure analysis, propose specific improvements to the memory operation bank.

Inputs.

- **Failure analysis:** `{analysis_feedback}`
- **Current operation bank:** `{operation_bank_full}`
- **Operation-evolution feedback:** `{evolution_feedback}`
- **Additional skill hint:** `{new_skill_hint}`

Task. Propose up to `{max_changes}` improvements based on the analysis:

- **Add a new operation** when the analysis reveals a capability gap.
- **Refine an existing operation** when an instruction template is too vague or misses key details.
- **Return no change** when failures are caused by retrieval issues or when the current operations are sufficient.

Critical requirements.

1. `instruction_template` must be a skill-style guide and must not include context placeholders.
2. `instruction_template` must clearly state purpose, when to use, and constraints.
3. `instruction_template` must specify target view, readable views, action scope, and allowed actions.
4. New operations must provide `target_view`, `source_views`, `action_scope`, `allowed_actions`, and optionally `supports_multi_action`.
5. `target_view` must be `graph`; allowed actions may include `ADD_ENTITY`, `MERGE_ENTITY`, `ADD_RELATION`, `UPDATE_RELATION`, `DELETE_GRAPH_OBJECT`, or `NOOP`.
6. Avoid labels such as "ENHANCED" or "ADVANCED"; keep descriptions neutral and task-specific.
7. Do not embed output blocks; the executor handles output formatting and can apply the skill multiple times.
8. The number of changes must be at most `{max_changes}`.
9. Do not modify the same operation more than once, and do not refine an operation added in the same response.

Example graph operation.

```
{
"name": "capture_family_relation",
"description": "Graph skill for capturing family
relationships mentioned in the current chunk.",
"target_view": "graph",
"source_views": ["chunk", "summary", "graph"],
"action_scope": "write",
"allowed_actions": ["ADD_ENTITY", "ADD_RELATION"],
"supports_multi_action": true,
"instruction_template": "Skill: Capture Family Relation
... Every new node or edge must cite SOURCE_CHUNK_IDS."
}
```

Output format. Respond with exactly one JSON object. For one or more changes, use:

```
{
"action": "apply_changes",
"summary": "<overall rationale>",
"changes": [
{
"action": "add_new",
"new_operation": {
"name": "<snake_case_name>",
"description": "<what it does and when to use it>",
"instruction_template": "<skill-style instruction
```

```
template>",
"target_view": "graph",
"source_views": ["<chunk | summary | graph>"],
"action_scope": "<write | maintain | hybrid>",
"allowed_actions": ["<ADD_ENTITY | MERGE_ENTITY |
ADD_RELATION | UPDATE_RELATION | DELETE_GRAPH_OBJECT |
NOOP>"],
"supports_multi_action": <true | false>,
"update_type": "<optional compatibility label>",
"reasoning": "<how this addresses the failures>"
}
}
]
}
```

For no changes, use:

```
{
"action": "no_change",
"reasoning": "<why the current operations are
sufficient>"
}
```

Instruction-template structure. Each instruction template should include:

```
Skill: <short skill name>
Target view: graph
Readable views: chunk, summary, graph
Purpose: <what this skill does>
When to use: <triggers>
How to apply: <steps>
Allowed actions: <permitted graph operations>
Constraints: <what to avoid>
```

Constraint. Output only the JSON object, with no additional text.

Graph Memory Construction

Role. You are a memory-management executor. Apply the selected skills to the input chunks and retrieved chunk, summary, and graph memories, then output structured graph actions.

Inputs.

- **Input chunks:** `{input_chunks}`
- **Retrieved memories:** `{retrieved_memories}`
- **Selected skills:** `{selected_skills}`

Guidelines.

- Apply only actions supported by the selected skills.
- Valid graph actions are `ADD_ENTITY`, `MERGE_ENTITY`, `ADD_RELATION`, `UPDATE_RELATION`, `DELETE_GRAPH_OBJECT`, and `NOOP`.
- Raw chunks are already stored by the system; do not output `STORE_CHUNK` or `UPDATE_SUMMARY`.
- For every `ADD_ENTITY`, `ADD_RELATION`, or `UPDATE_RELATION` action, `SOURCE_CHUNK_IDS` must be selected only from the allowed `SOURCE_CHUNK_IDS`.
- Do not invent identifiers such as `chunk_1`, `chunk_2`, `graph:xxx`, `summary:D2`, `summary:D18`, or `retrieved_memory_7`.
- Prefer 4–5 directly supporting chunk IDs when a fact is supported across multiple input chunks, and never use more than 5.
- If a graph write cannot be supported with allowed local chunk IDs, return `NOOP` instead of inventing IDs.

Constraint. Output only action blocks. Do not include explanations outside the blocks.

Question-Answering

Role. 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.

Critical rules.

- **Semantic reasoning for non-temporal questions.** When answering questions about preferences, habits, or facts, apply semantic reasoning: "X is my weakness" means loves or favorite X; "brings joy" and "really like" mean loves; "I limit myself to X per day" means loves X.
- **Time conversion for temporal questions.** When answering questions about when something happened, never output relative time expressions such as "last week", "yesterday", "tomorrow", or "last Friday". Always convert the reference to a specific date and show the calculated date in the answer.

Instructions.

1. Carefully analyze all provided memories from both speakers.
2. Pay special attention to 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. For questions about time references, calculate the actual date based on the memory timestamp.
6. Always convert relative time references to specific dates, months, or years.
7. Focus only on the content of the memories from both speakers.
8. Apply the semantic reasoning rules before concluding that information is missing.
9. Keep the answer under 5–6 words.

Inputs.

- **Relevant memories:** `{context_memories}`
- **Question:** `{question}`

Output. Answer the question directly.

Gating Prompt

Task. Filter memories for a multi-fact integration query with complexity level 1. Use balanced filtering: keep facts that contribute to the answer.

Target. Keep 5–12 memories.

Filtering rules.

1. Keep a memory if it contains facts relevant to the question.
2. Keep a memory if it provides temporal or contextual connections.
3. Keep a memory if it helps establish patterns or sequences.
4. Exclude a memory if it is only tangentially related.
5. Exclude a memory if it is redundant with other kept memories.

Inputs.

- **Question:** `{question}`
- **Candidate memories:** `{numbered_memories}` among `{total_count}` total memories.

Output format. `{ "relevant_ids": [1, 2, 3, ...] }`

F POLICY-SELECTOR ABLATION

We compare the PPO dual-encoder controller with an LLM-as-policy controller on the LoCoMo test split. Both selectors use the same fixed Epoch-6 skill library, input state, selection top-k, and downstream skill executor. The comparison changes only the skill selector, separating policy selection from library evolution.

Table 10: Policy-selector comparison with the Epoch-6 skill library held fixed.

Selector	F1 ↑	BLEU-1 ↑
PPO controller	**0.517933**	**0.425141**
LLM-as-policy controller	0.506788	0.407840
Difference (LLM-as-policy − PPO)	−0.011145	−0.017301

Replacing the PPO controller with the LLM selector decreases F1 by 0.011145 and BLEU-1 by 0.017301. Under this fixed-library setting, the learned controller selects more effective memory operations. This experiment complements the initial-library ablation in the main text: that comparison evaluates library evolution while retaining PPO policy learning, whereas the present comparison evaluates the selector with an evolved library held fixed. The two experiments therefore provide separate evidence for operator induction and policy selection.

G Additional Evaluation

G.1 Full-History Question Answering

We compare 3D-Mem with direct full-history prompting using the same answer backbone, QA prompt, and decoding settings. Table 11 reports F1 on the LoCoMo test split (314 questions) and LongMemEval-S (500 examples). The 3D-Mem scores match Tables 2 and 3.

Table 11: F1 comparison with direct full-history QA. Δ denotes 3D-Mem minus full-history. N/A indicates that complete histories exceed the serving configuration's context limit.

Answer backbone	Dataset	3D-Mem	Full-history	Δ
Qwen3-30B-A3B	LoCoMo	**0.5179**	0.4138	+0.1041
Qwen3-30B-A3B	LongMemEval-S	0.5220	N/A	—
LLaMA3.3-70B-Instruct	LoCoMo	0.5029	**0.5143**	−0.0114
LLaMA3.3-70B-Instruct	LongMemEval-S	**0.4616**	0.1981	+0.2635

On LoCoMo, 3D-Mem improves F1 by 0.1041 with Qwen, whereas full-history prompting leads by 0.0114 with LLaMA. In the latter comparison, full-history QA uses approximately 22K input tokens per query versus approximately 2K for 3D-Mem. These counts refer to total query input, not the retrieved-evidence counts in Table 4. On LongMemEval-S, histories average approximately 105K tokens; the Qwen serving configuration is limited to 40,960 tokens, preventing complete-history evaluation without truncation. LLaMA accommodates the full history, but 3D-Mem improves F1 by 0.2635. These results support a quality–context trade-off rather than uniform accuracy superiority over full-history prompting.

G.2 Validation Assessment of Evolved Checkpoints

Table 12 evaluates checkpoints from all nine outer epochs on LoCoMo validation conversations 7–8. We select Epoch 6 solely by validation overall F1: it attains the highest value, 0.5789, among these nine checkpoints. Test scores are not used in this selection rule.

Table 12: Validation overall F1 for the evaluated checkpoints.

Epoch	1	2	3	4	5	6	7	8	9
F1	0.5494	0.5630	0.5410	0.5482	0.5433	**0.5789**	0.5414	0.5450	0.5536

Operator additions, refinements, and removals can change memory granularity, so the evolution procedure does not guarantee monotonic validation improvement. Appendix D illustrates how these updates change the specificity of retrieved evidence.

G.3 INDEPENDENT TRAINING RUNS

We repeat PPO training independently three times with different random seeds using Qwen3-30B-A3B. Table 13 reports F1 and BLEU-1 for each run and their mean and sample standard deviation, computed with denominator $n - 1$ for $n = 3$. Run 1 denotes the same Qwen-trained controller and operator-library checkpoint evaluated on both datasets; its scores match the main tables. The other run indices identify additional independent training results within each dataset and do not assert cross-dataset pairing. These results characterize training variability; the main tables retain individual-run comparisons.

Table 13: Qwen results across three independent PPO training runs. Scores are on a 0–1 scale; SD denotes sample standard deviation, not standard error.

Dataset	Run	F1	BLEU-1
LoCoMo	1	0.517933	0.425141
	2	0.519116	0.426803
	3	0.502554	0.418123
	Mean $\pm$ SD	0.513201 ± 0.009240	0.423356 ± 0.004607
LongMemEval-S	1	0.522000	0.425200
	2	0.505274	0.404969
	3	0.502171	0.402237
	Mean $\pm$ SD	0.509815 ± 0.010666	0.410802 ± 0.012544

The F1 standard deviation is approximately 0.0092 on LoCoMo and 0.0107 on LongMemEval-S. These estimates summarize the observed runs; they do not establish statistical significance against baselines. In particular, the LoCoMo test set contains only two conversations, so its 314 questions should not be treated as independent conversation-level replicates.