CUSTOMIZE YOUR MEMORY: SCENARIO-ADAPTIVE AND EVOLVABLE MEMORY MODELING FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory enables large language model (LLM) agents to learn continually across sessions and scenarios. Memory systems have attracted great attention as LLM agents are widely used in coding, office work, and other scenarios. However, memory modeling, i.e. how to represent flat text and what features to extract, is either ignored or hand-crafted and frozen at design time. We argue that memory is a process of identification, extraction, and synthesis, i.e. an offline cognitive capability whose core is memory modeling instead of a flat archive. We formulate the memory system as a modeling-guided optimization problem and present ScenarioModelMem, a scenario-adaptive memory-modeling system. We split memory modeling into a static memory structure and a dynamic memory policy to achieve efficient representation, flexible configuration, and optimizability. A validation-driven evolutionary algorithm refines the dynamic memory policy through error-informed mutation, crossover, and pruning. Under the same experimental configuration as the strongest published systems, ScenarioModelMem achieves state-of-the-art performance on LOCOMO (94.03) and PersonaMem (70.63) with a pre-defined memory policy. Furthermore, we evaluate automatic policy design on PersonaMem. Starting from a basic initial policy, the evolved policy outperforms a carefully crafted human-designed one. Memory modeling thus becomes scenario-adaptive and optimizable, rather than a fixed design choice.

1 INTRODUCTION

From personal assistants to autonomous analysts, LLM agents increasingly operate in long-horizon, knowledge-intensive workflows whose success depends less on reasoning over the current context than on what the agent remembers across time and scenarios (Packer et al., 2023; Zhong et al., 2024; Chhikara et al., 2025). Over the past few years, memory has become one of the most important factors that determine agent capability and test-time learning performance (Hu et al., 2025b; Zhou et al., 2026). Existing work falls into three categories. The first is memory operating systems (Packer et al., 2023; Li et al., 2025; Kang et al., 2025), which are among the most successful and widely deployed designs in agent harnesses. Their predecessor is retrieval-augmented generation (RAG), which augments models and agents with external knowledge bases (Lewis et al., 2020; Edge et al., 2024; Asai et al., 2024). Factual-memory examples include Mem0, MemOS, EverMemOS, and Mandol (Chhikara et al., 2025; Li et al., 2025; Hu et al., 2026; Zhang et al., 2026b); experience-memory examples include ExpeL and ReasoningBank (Zhao et al., 2024; Ouyang et al., 2026). Recent work further studies multi-granularity association and selection of long-term memory (Xu et al., 2026).

The second category is memory models. Beyond explicit textual memory, two memory types are of particular interest: latent memory and parametric memory; both operate through model inference, which encodes and decodes information. Latent memory is stored as cached latent embeddings and acts at the sequence level, as in the KV cache (Wu et al., 2022), MemoryLLM (Wang et al., 2024), and M+ (Wang et al., 2025). Parametric memory generally acts on attention layers or linear layers influencing model weights and policy. LoRA-style adapters are a common form (Hu et al., 2022), but they are fixed after training. Other architectures such as Titans (Behrouz et al., 2024) and fast-weight product-key memory layers (Zhao & Jones, 2026) follow the test-time-training method (Sun et al.,

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.

2024). Test-time-training layers compute gradients on fast parameters at inference time, so memory is updated online rather than frozen. Compared with memory systems, memory models optimize the memory strategy end-to-end through model training. However, this line of work requires large amounts of long-context training data. The cost in compute and time is high. Multi-user isolation also remains difficult.

The third category is post-trained memory strategies. Memory-R1 trains the policies for memory generation, consolidation, and retrieval-augmented answering (Yan et al., 2025). Memento learns a recall policy through lightweight kernel-based soft Q-learning (Zhou et al., 2025). MemRL builds and learns Q functions to modulate recall scores (Zhang et al., 2026a). Recent work also trains the model itself to manage memory through reinforcement learning (Yu et al., 2026; Du et al., 2026). In essence, this line of work keeps the memory system structure unchanged and optimizes only the strategies through training. Its gains are most evident when the data distribution is fixed.

Among the three categories, memory systems are the most widely used and the most mature in practice. However, we find that most of them focus on the structural organization of memory, such as flat memory (Chhikara et al., 2025), tree-structured memory (Sarthi et al., 2024), and graph-structured memory (Edge et al., 2024; Jiménez Gutiérrez et al., 2024). They focus far less on joint memory modeling, that is the design of memory content, structure and data relations. To show the importance of memory modeling, we propose ScenarioModelMem, a scenario-adaptive memory-modeling system. It reconstructs information as a discrete three-dimensional space over entities, properties, and time. A domain-specific, optimizable memory policy guides the mapping. The representation is easy to extend and can be tailored to a specific scope of information. Combined with a dynamic 3D memory search engine, ScenarioModelMem achieves state-of-the-art accuracy among memory systems without using oracle information for reflection.

Our core contributions are threefold:

- We design ScenarioModelMem, a memory system centered on memory modeling. Its store is a 3D memory graph over entity, property, and time. Two tailored operators serve the store: memory generation and dynamic memory search. Under the same experimental configuration as the strongest published systems, ScenarioModelMem achieves state-of-the-art accuracy.
- For flexible scenario adaptation, to our knowledge we are the first to split memory modeling into a static memory structure and a dynamic memory policy. The split decouples the hard configuration (the structure) from the soft configuration (the policy). Memory modeling thus becomes optimizable. We identify the memory policy as the key optimization variable.
- We design an evolutionary search algorithm for memory modeling. It designs the memory policy fully automatically. The evolved policy outperforms a carefully crafted human-designed policy in our experiments.

2 Related work

In this section, we review existing work on retrieval-augmented generation (RAG), memory systems, and LLM-guided heuristic optimization related to our design.

2.1 Retrieval-augmented generation

Retrieval-augmented generation (RAG) is the direct predecessor of memory systems (Lewis et al., 2020). It retrieves relevant text through keyword or vector search and injects it into the prompt dynamically. Follow-up studies optimized each stage of the pipeline in detail: chunking, reranking (Sun et al., 2023), and query rewriting (Ma et al., 2023) or expansion (Gao et al., 2022). The text organization also evolved from flat text to trees (Sarthi et al., 2024) and graphs (Edge et al., 2024; Guo et al., 2025). Agentic RAG performs adaptive multi-round retrieval (Asai et al., 2024; Jeong et al., 2024). The retrieval policy can even be trained end-to-end with reinforcement learning (Jin et al., 2025). Throughout this evolution, the optimization variable is the retrieval pipeline. The knowledge itself remains raw text. How the knowledge is modeled is left open.

2.2 MEMORY SYSTEMS

Memory systems have become a major component of agent harnesses. Compared with RAG, they focus on memory extraction, memory integration, and offline consolidation. Broadly, existing systems fall into two branches: factual memory and experience memory. On the factual side, Mem0 is arguably the founding work (Chhikara et al., 2025). It adopts minimalist fact extraction. It defines the core memory operations: merge, update, add, and delete. Its optimized variants achieve strong scores through multi-route retrieval. Some systems, such as Mem0^g and Zep (Chhikara et al., 2025; Rasmussen et al., 2025), organize memory as graphs; similar to GraphRAG, they model relations among memories and answer multi-hop questions efficiently. MIRIX manages six types of memory through multi-agent collaboration (Wang & Chen, 2025), and EverMemOS organizes MemCells into thematic MemScenes (Hu et al., 2026). Mandol aggregates memories into hierarchical semantic graphs (Zhang et al., 2026b). Memora balances abstraction and specificity (Xia et al., 2026).

The other branch is experience memory: these systems summarize and analyze the policies an agent has executed, distilling knowledge, policies, and lessons for later similar tasks. ExpeL is an early representative (Zhao et al., 2024). ReasoningBank stores reasoning strategies from both successes and failures (Ouyang et al., 2026). SCOPE manages guidelines at the tactical and the strategic level (Pei et al., 2025). Across both branches, the modeling is designed by hand and frozen at design time, whether the structure is flat entries, graphs, or layered hierarchies. Some systems evolve memory organization at run time (Xu et al., 2025), and EvolveMem even searches over retrieval-side configurations (Liu et al., 2026). Yet the modeling policy itself never enters the optimization loop. In contrast, we make the modeling policy dynamic and turn it into an optimization variable.

2.3 LLM-GUIDED HEURISTIC OPTIMIZATION

The earliest search target for text optimization was the prompt. OPRO first showed that an LLM can act as an optimizer (Yang et al., 2024). At each step, the LLM reads previous solutions and their scores, and proposes new ones. EvoPrompt connects this idea with evolutionary algorithms: the LLM plays the mutation and crossover operators, and prompts evolve as a population (Guo et al., 2024). Promptbreeder makes the evolution self-referential, so the mutation prompts evolve together with the task prompts (Fernando et al., 2024). FunSearch then took the search into program space and produced new mathematical discoveries (Romera-Paredes et al., 2024). EoH makes program search sample-efficient by evolving natural-language thoughts and code together (Liu et al., 2024). ReEvo adds a reflective gradient in natural language (Ye et al., 2024). The search target then expanded from prompts and programs to whole pipelines. DSPy compiles declarative calls into optimized pipelines (Khattab et al., 2024). TextGrad backpropagates textual feedback through them (Yuksekgonul et al., 2025). AFlow searches the workflow space with tree search (Zhang et al., 2025). ADAS lets a meta agent program whole agents in code space (Hu et al., 2025a).

Our optimizer inherits its machinery from this line of work: validation-driven, LLM-guided evolution over a population of discrete objects. Elites are preserved. Selection, crossover, and mutation generate the next generation. Validation errors inform the mutation. We aim this machinery at the memory policy. The policy shapes the store at memory generation time.

3 METHOD

3.1 PROBLEM FORMULATION

First, we formalize a general memory system. The input is a message stream $\mathcal{M} = (m_1, m_2, \ldots)$ that grows across sessions. The system maintains a memory state G_t, the accumulated store at time t. Memory modeling determines the mechanism: how memories are extracted and distilled from information, and how they are organized. We split it into two orthogonal parts, a static structure Σ and a dynamic policy π. The structure Σ is the data format and structure of G_t. The policy π is the dynamic part that varies across scenarios. It determines what content deserves attention and how information is organized and associated.

Two operators form the basic execution functions of a memory system. Add is the write path: it generates memories from the new message and updates the state. Search is the read path: it returns

the records of the state relevant to a query q. We define the two operators as A and R. Formally,

$$G_t = A(G_{t-1}, m_t;\, \Sigma, \pi), \qquad R(q; G_t;\, \Sigma, \pi), \tag{1}$$

where the execution of both operators follows the memory modeling: the structure Σ as the form of the state they act on, and the policy π as the guidance they follow.

The design goal of a memory system is explicit: choose Σ and π so that the state best organizes the information and provides reasoning-supporting evidence for queries. Beyond plain information access and retrieval, the memory system also completes or shortens part of the reasoning chain for some queries. For typical QA queries, this goal is formalized as

$$(\Sigma^*, \pi^*) \;=\; \arg\max_{\Sigma,\,\pi} \;\; \mathbb{E}_{(q,\,a)} \big[\mathrm{Judge}\big(a,\; \mathrm{LLM}\big(q \,\big|\, R\big(q;\; A(\mathcal{M};\, \Sigma, \pi);\, \Sigma, \pi\big)\big)\big)\big] \,, \tag{2}$$

where the state $G_{\Sigma,\pi} = A(\mathcal{M};\, \Sigma, \pi)$ is built by folding the Add of Eq. 1 over the scenario's message stream $\mathcal{M}$ from an empty store, $\mathrm{Judge}(a, \hat{a}) \in [0, 1]$ scores the generated answer against the reference, and the answer model stays frozen. Sections 3.3 to 3.5 introduce Σ and both operators; Section 3.6 turns the policy into a searchable objective.

3.2 System overview

ScenarioModelMem instantiates this formalization as shown in Figure 1. A memory structure is typically an enumerable, finite design variable, yet few existing methods search and optimize it. We thus aim at a general and highly expressive structure, universal across scenarios, so that adapting to a new one only requires updating the policy. To this end, we fix the structure Σ as a 3D graph over entity, property, and time. The policy π is a declarative manifest: it states the entity types to remember and the properties each entity carries. Correspondingly, we design fine-grained generation and search pipelines that maintain a high-quality state and exploit it fully to improve recall. The system consists of three modules. Module I is the policy-guided generation pipeline. Module II covers memory modeling and storage. Module III is dynamic search.

3.3 Scenario-adaptive memory modeling

As described in Section 3.1, ScenarioModelMem introduces the concept of memory modeling and splits it into a static, universal memory structure and a scenario-customized memory policy. For the structure, we target strong representational power and the ability to associate, present, and summarize knowledge. We design Σ as a three-dimensional space, i.e. entity, property, time. Memory is organized around entities. Every object that occurs in the messages is modeled as an entity and described by a set of property categories, i.e. discrete slots, not free text. Each property carries a time-ordered sequence of values rather than a single latest one. This design maintains a complete profile for every entity, whereas mainstream memory systems center on the user profile alone. Preserved history and per-property timelines keep each entity's development traceable. Relations among entities form a relation network, as in mainstream graph-structured memory. Each concrete property becomes an atomic memory at a coordinate (e, p, t), so that facts, relations, persona profiles, and temporal sequences live in one addressable, versioned structure. The store G_π under this instantiation is the set of these entity records plus two kinds of links: semantic edges between entities, and temporal edges between successive values of the same entity-property pair.

The policy π is a manifest over this space: it declares the entity types to remember and the properties each type carries. *Primitive* properties record the explicit facts stated in the text, without further inference. *Higher-order* properties are inferred by synthesizing patterns across multiple observations, such as a user's decision-making style. Changing π changes both what the store contains and what the search operators retrieve, while the structure Σ stays the same. The policy is thus the part of the modeling that varies across scenarios, and a well-defined, optimizable variable (Section 3.6).

3.4 Policy-guided memory generation

The generation pipeline turns the continuous message stream into discrete records at coordinates through four stages guided by π: episode segmentation, memory generation, entity fusion, and graph merge.

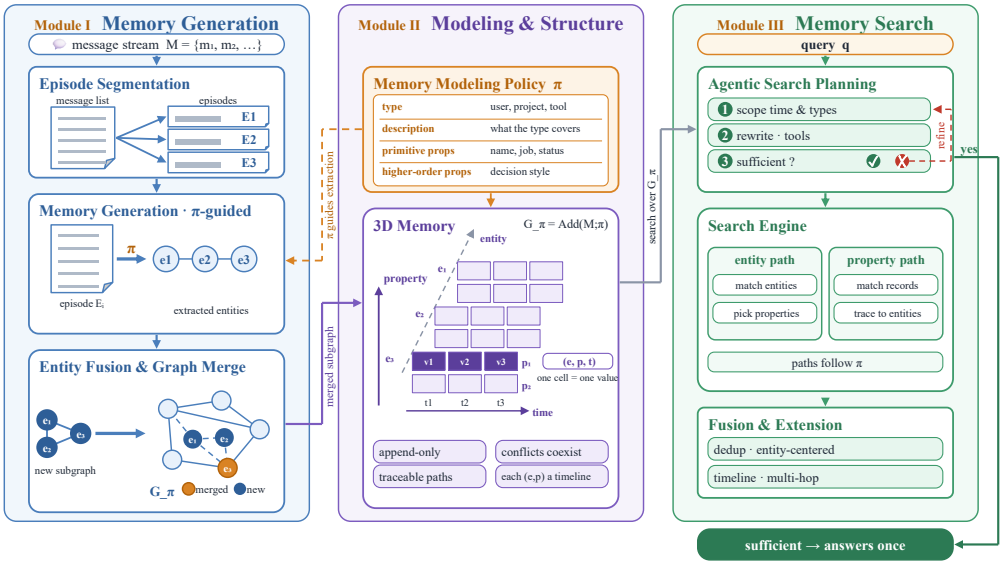


Figure 1: Overview of ScenarioModelMem and its three modules: the policy-guided generation pipeline (Module I), memory modeling and storage (Module II), and the dynamic search (Module III).

The pipeline starts with episode segmentation: an LLM segmenter partitions the incoming messages into complete episodes, each covering one sub-topic, independent of message roles and session boundaries.

Memory generation then maps each episode onto the entity and property types declared by π. It extracts entity instances and their property values, resolving temporal expressions into absolute timestamps when the evidence suffices. We include one *episodic entity* per episode as a fallback, preserving context that the policy does not capture.

Entity fusion anchors the extracted subgraph to the store G_π. First, the subgraph serves as a query for retrieving related memories and resolves entity equivalence. Each match is merged into the existing entity: descriptions are updated, property values are inserted or revised along their timelines, and overlapping values are consolidated. Higher-order properties declared by π are checked and synthesized from the relevant history.

Finally, graph merge fuses the nodes and forms new edges, merging the subgraph into the store.

3.5 Dynamic Memory Search

The outer layer of the retrieval module in ScenarioModelMem is agentic search. An LLM controller plans retrieval iteratively: it sets the temporal scope and memory types, formulates queries, invokes the search engine, and either refines the plan or terminates once the evidence suffices.

The engine exposes two search directions: the forward direction matches entities first and then selects their relevant properties ($e \rightarrow p$); the reverse matches property-level records first and then traces them to their parent entities ($p \rightarrow e$). Both fuse BM25 and dense embedding vector similarity rankings by reciprocal rank fusion (RRF), and results from the two directions are deduplicated into

an entity-centered view. Retrieved memories extend temporally through predecessor and successor versions of a property; relational expansion through associated entities is also conducted.

To imporve recall rate, we use search field augmentation for each entity, fine-grained matching expands an entity description into multiple search fields; the highest query-field matching score becomes the entity score improving detail information match.

3.6 MEMORY EVOLUTION

The memory policy (π) is the soft variable controlling system behavior. It is scenario-dependent: an agent needs different entity types and properties in different task categories. Remembering everything not only raises cost, but also introduces more distractor information downstream. The policy is a searchable configuration (Appendix C)

This optimizes the design goal of Eq. 2 with Σ fixed and the policy π as the only free variable. The policy is optimized offline on a fitting set of N samples, $\mathcal{D} = \{(c_i, q_i, a_i)\}_{i=1}^{N}$, each pairing a context c_i with a query q_i and its reference answer a_i. For a candidate policy π, the fitting contexts are added under π, the queries are answered based on dynamic search results (Section 3.5). The answers are scored by the Judge. We select the policy that maximizes a fitness $f(\pi)$, the mean Judge score under π minus a size penalty on the memory state,

$$\pi^* \;=\; \arg\max_{\pi} \;\; \frac{1}{N}\sum_{i=1}^{N} \mathrm{Judge}\big(a_i,\; \mathrm{LLM}\big(q_i \,\big|\, R\big(q_i; G_\pi\big)\big)\big) \;-\; \lambda\,|G_\pi|, \tag{3}$$

where $G_\pi = A(\mathcal{C};\, \pi)$ is the store built by folding the Add of Eq. 1 over the fitting contexts $\mathcal{C} = \{c_i\}_{i=1}^{N}$ under the policy π, and $\mathrm{Judge}(a, \hat{a}) \in [0, 1]$ scores the generated answer against the reference. The term $\lambda\,|G_\pi|$ is a size regularizer: $|G_\pi|$ is the number of records in the store, and $\lambda \geq 0$ sets the trade-off between accuracy and state size. A smaller state costs less to maintain and leaves fewer distractors for search.

The samples are grouped into U clusters. The algorithm runs E epochs over the clusters, $H = E \cdot U$ generations, and maintains a population of K policies; $\pi_{h,j}$ denotes the j-th individual at generation h.

In each generation, the active cluster supplies the evaluation scope. For every individual $\pi_{h,j}$, a sandboxed instance with a clean store ingests the cluster contexts, answers the cluster queries through dynamic search, and evaluates the fitness $f(\pi_{h,j})$ of Eq. 3 (the full loop in Appendix A).

The top α fraction of individuals are preserved as elites; the remaining slots are filled with offspring of parents drawn by tournament selection. Error-informed mutation summarizes the parent's evaluation errors for the LLM, which locates information gaps and proposes new entity types, new properties, or refined descriptions. Exploratory mutation employs the LLM to inject plausible types for the target scenario and prunes low-frequency elements with probability p_{prune}. Crossover uses one parent as the base policy and inserts entities and properties from the other with probability p_{cross}.

After H generations, all individuals of the final population are evaluated on the full fitting set. The best one, π^*, is selected. Evolution is offline learning: its output is a new system configuration, not an update to a running store.

4 EXPERIMENTS

4.1 MEMORY SYSTEM ACCURACY

We first evaluate ScenarioModelMem without memory evolution on two commonly used long-term-memory benchmarks: LOCOMO and PersonaMem. LOCOMO (Maharana et al., 2024) assesses information extraction, effective recall, and simple compositional reasoning over multi-memory associations. We focus on its four main reasoning types: Single-hop, Multi-hop, Temporal, and Open-domain. PersonaMem (Jiang et al., 2025) examines user profiling, preference analysis, and style-aware personalized response. It contains 20 personas with 37 independent context scopes and seven query types: Recall Sha. (recalling user-shared facts), Recall Mem. (acknowledging the user's

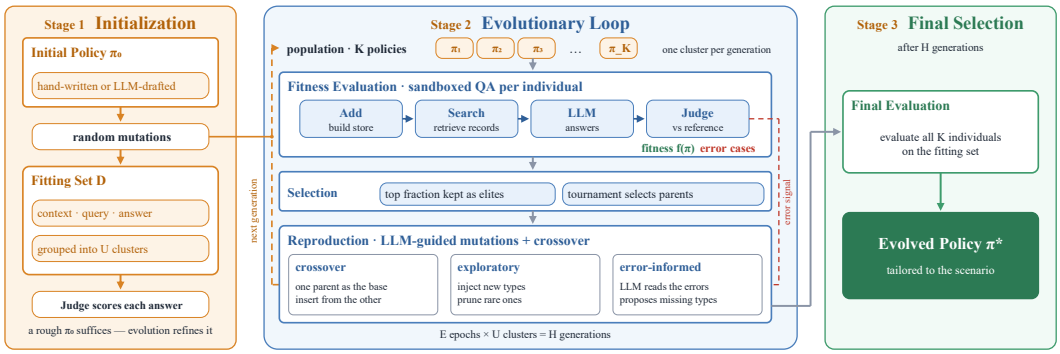


Figure 2: Validation-driven evolutionary policy search.

Table 1: QA accuracy (%) on LOCOMO by reasoning type under two answer models.

	gpt-4o-mini					*gpt-4.1-mini*				
Method	Single	Multi	Temp.	Open	Overall	Single	Multi	Temp.	Open	Overall
Mem0	66.71	58.16	55.45	40.62	61.00	68.97	61.70	58.26	50.00	64.20
memU	72.77	62.41	33.96	46.88	61.15	74.91	72.34	43.61	54.17	66.67
MemOS	81.45	69.15	72.27	60.42	75.87	85.37	79.43	75.08	64.58	80.76
Zep	88.11	71.99	74.45	66.67	81.06	90.84	81.91	77.26	75.00	85.22
EverMemOS	91.08	86.17	81.93	66.67	86.76	96.67	91.84	89.72	76.04	93.05
ScenarioModelMem	**93.47**	84.40	**83.80**	64.58	**87.99**	**96.79**	**93.97**	**90.34**	**82.29**	**94.03**

latest preferences), Track Evo. (tracking full preference evolution), Revisit (revisiting reasons behind preference updates), Suggest. (suggesting new ideas), Recom. (providing preference-aligned recommendations), and General. (generalizing to new scenarios).

We compare against mainstream memory-system baselines: Mem0 (Chhikara et al., 2025), memU (NevaMind-AI, 2025), MemOS (Li et al., 2025), Zep (Rasmussen et al., 2025), and EverMemOS (Hu et al., 2026), following the EverMemOS evaluation protocol. The LLM driving the memory system, the embedding and reranking models, the answer model, and the judge model are fully aligned with the EverMemOS default implementation; on LOCOMO the judge is gpt-4o-mini with the benchmark's unified rubric prompt. We evaluate our memory system under both gpt-4o-mini and gpt-4.1-mini; we reproduced EverMemOS on LOCOMO with broadly matching results, so baseline results in both settings are cited from the EverMemOS paper for uniformity. On PersonaMem, baselines are evaluated with local deployments of their open-source code under the same configuration, with gpt-4.1-mini as the answer model. We report per-category accuracy (%) together with the overall accuracy.

Table 1 shows the LOCOMO results. ScenarioModelMem achieves the highest overall accuracy in both settings: 94.03 under gpt-4.1-mini and 87.99 under gpt-4o-mini, ahead of EverMemOS (93.05 and 86.76). Under gpt-4.1-mini, it shows particular strength in Single-hop (96.79) and Multi-hop (93.97) reasoning and achieves the highest Open-domain score (82.29). This comes from the full-information modeling of the 3D structure and the two-direction, multi-source retrieval of dynamic search as discussed in ablation study. LOCOMO contains mislabeled questions and questions hinging on complex model reasoning; a memory system alone can hardly fix them, and the current gain is already considerable. On PersonaMem (Table 2), the policy declares the user as the central entity and carries common higher-order properties for in-depth user profiling. ScenarioModelMem achieves the best overall accuracy (70.63), a gain of 3.06 percentage points over EverMemOS (67.57), which shows the advantage of modeling higher-order properties. The gain is driven primarily by Suggest. (47.31 vs. 35.48), Recom. (76.36 vs. 65.45), and Recall Sha. (81.40 vs. 74.42). The Recall Mem. category contains only 17 questions; one or two correct answers can move its accuracy substantially.

Table 2: Accuracy (%) on PersonaMem by query type.

Method	Recall Sha.	Recall Mem.	Track Evo.	Revisit	Suggest.	Recom.	General.	Overall
Mem0	46.51	41.18	65.47	90.91	12.90	34.55	43.86	51.61
memU	64.34	64.71	66.20	87.88	31.18	67.27	84.21	65.70
MemOS	53.49	82.35	66.91	79.80	41.94	69.09	75.44	63.67
EverMemOS	74.42	64.71	64.03	85.86	35.48	65.45	84.21	67.57
ScenarioModelMem	**81.40**	64.71	64.75	82.83	**47.31**	**76.36**	73.68	**70.63**

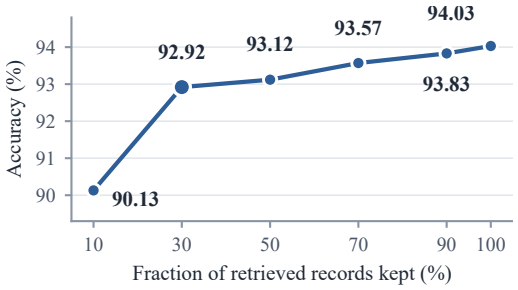


Figure 3: Accuracy vs. the fraction of retrieved records kept per question on LOCOMO. The retrieved records are reranked before truncation; 100% is the untruncated system.

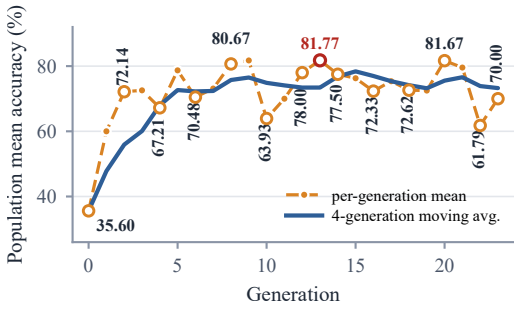


Figure 4: Policy evolution on PersonaMem: population mean accuracy per generation (dashed) and its 4-generation moving average (solid). Each generation is evaluated on one fitting-set cluster.

4.2 Ablation Study

We now ablate two aspects of ScenarioModelMem: (1) the system structure, and (2) the relation between memory quantity and accuracy.

For the system structure, we ablate the two core components on LOCOMO (Table 3). The reference is the full system (gpt-4.1-mini) of Table 1. The first variant removes the 3D memory structure and falls back to flat episodic factual memory; the generation pipeline then runs without detailed policy design and guidance. The second variant removes the dynamic search engine and falls back to plain reciprocal-rank-fusion retrieval over the entity library. The two removals drop the overall accuracy from 94.03 to 91.82 and 92.27, respectively. Without the 3D memory structure, memory is no longer managed along entity, property, and timeline. Joint recall of associated information suffers at retrieval time, and the evidence becomes fragmented at answer time. Accuracy on both retrieval-intensive and Open-domain queries drops sharply (Multi-hop 93.97 to 89.01; Open-domain 82.29 to 75.00). Without dynamic search, retrieval degenerates to one-pass global entity search. Agentic planning, iterative rounds, and the reverse path from properties to entities are all missing. Overall recall drops accordingly, and the drop spreads more evenly across query types.

For the memory quantity, we study how much retrieved memory the answers actually need. The starting point is the complete LOCOMO run of Table 1 (94.03). We collect the full retrieval memories of every question, rerank them with a Qwen3 reranker (4B), and truncate them to the top 10%, 30%, 50%, 70%, and 90%. The questions are then re-answered with gpt-4.1-mini, and accuracy is computed again. Figure 3 plots accuracy against the fraction of retrieved records kept. Keeping 30% of the records already yields 92.92. On average, each question then involves 6.83 entities containing 11.80 properties in total. The remaining 70% adds little for most questions; only a few hard cases keep relevant records in the tail. A small, well-ranked portion of the retrieved records thus carries most of the evidence.

Table 3: Ablation study on LOCOMO (%).

Method	Single-hop	Multi-hop	Temporal	Open-domain	Overall
ScenarioModelMem (full)	**96.79**	**93.97**	**90.34**	**82.29**	**94.03**
w/o 3D memory structure	96.20	89.01	87.85	75.00	91.82
w/o dynamic search (RRF)	95.96	91.13	87.85	78.12	92.27

4.3 Effectiveness of memory evolution

We now evaluate automatic policy design on PersonaMem: the evolution algorithm searches for a good policy without human intervention. We split PersonaMem at the scope level, randomly selecting 12 scopes from 7 personas as the fitting set, 196 questions in total. The remaining 393 questions form the test set. Context scopes and questions are disjoint between the two sets. PersonaMem queries are objective multiple-choice questions; the Judge reduces to deterministic rule-based scoring, so evolution fitness and reported accuracy share one objective scorer. The evolution is guided by gpt-5.5; Figure 4 plots its population mean accuracy over generations. The mean jumps to 72.14 within two generations and then oscillates between 61.79 and 81.77. The oscillation follows the evaluation protocol: each generation uses a different active cluster, whose difficulty varies. The 4-generation moving average stays between 72.24 and 78.39 after generation 5 and ends at 73.25. Figure 5 compares four policy configurations under the same models and search configuration. To isolate the policy effect, it removes the catch-all default property and the episodic entity fallback, as in a practical deployment, and switches the answer model to gpt-5.5, so reasoning capability does not mask policy differences. During the final evaluation, we further insert four irrelevant books into the context as distractors, each truncated to the length of one PersonaMem scope. The raw baseline runs the system with the bare initial policy. The human-designed general policy and the human-designed PersonaMem policy are the pre-defined policies of Tables 1 and 2, respectively. The evolved policy is the output of Section 3.6. The evolved policy scores 75.32 on the test set and 74.36 overall (Figure 5). It outperforms both human-designed policies on the test set (73.79 and 72.77) and overall (73.34 and 72.50). Its fitting score matches that of the human-designed PersonaMem policy (72.45), so the test-set gain does not come from in-sample tuning. Among the three designed policies, the evolved policy keeps the smallest store (Figure 5): 42.21k characters per scope, versus 64.13k and 51.93k. The raw baseline keeps only 5.46k, yet its overall accuracy collapses to 47.88 under the distractors.

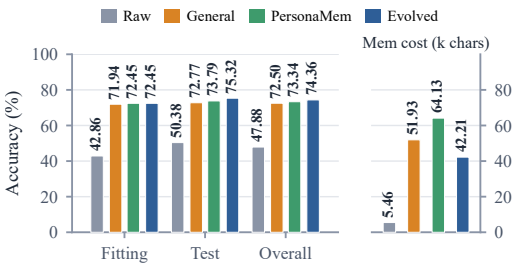


Figure 5: Memory evolution on PersonaMem: fitting, test, and overall accuracy (%) of the four policy configurations, and the memory cost per scope (k characters).

5 Conclusion

We presented ScenarioModelMem: a 3D memory graph over entity, property, and time, served by memory generation and dynamic search. Under the same configuration as the strongest published systems, it achieves state-of-the-art accuracy on LOCOMO (94.03) and PersonaMem (70.63) with a pre-defined policy. We split memory modeling into a static structure and a dynamic policy, decoupling the hard from the soft configuration; the declarative policy becomes a well-defined, optimizable variable. A validation-driven evolutionary algorithm then designs the policy fully automatically; on PersonaMem, the evolved policy outperforms a carefully crafted human-designed one. For memory evolution, our contribution is this decoupled design and the optimizable policy variable; the evolutionary algorithm is one instantiation, and text-gradient or other optimizers may also drive it. Comparing optimizers is not our focus. Evolution requires labeled fitting queries and an offline cost. Future work will focus more on optimization methods and online continual optimization.

AI USE STATEMENT

In this work, we used generative AI tools (Claude, Anthropic) for the following tasks. Required disclosure: assist with translation of author-written drafts. Recommended disclosure: edit the paper for readability, check grammar and key information, verify references against their sources, and write the figure-generation code. All research ideas, method design, and experimental results originate from the authors. The content, data, and layout of every figure and table were specified and verified by the authors. We have reviewed all AI-assisted work and take responsibility for the final content of this work.

ETHICS STATEMENT

This work evaluates memory systems on the public benchmarks LOCOMO and PersonaMem. It involves no human subjects and no data collection beyond these public corpora. Agent memory systems store and retrieve conversation content by design. Deploying them for real users therefore raises privacy considerations. Practitioners should obtain user consent and comply with data-protection regulations before storing personal information.

REPRODUCIBILITY STATEMENT

The method is fully specified in the main text and appendices: the memory structure (Section 3.3), the generation pipeline (Section 3.4), the dynamic search engine (Section 3.5), and the evolution algorithm with its fitness (Section 3.6, Eq. 3). Appendix A gives the full pseudocode of memory generation, dynamic search, and the evolution loop; Appendix B shows a small store example; and Appendix C lists the initial, pre-defined, and evolved policy manifests. Section 4.1 states the evaluation protocol, model configuration, and baselines; Section 4.3 states the fitting split and the distractor protocol. Complete manifests, source code, and the evaluation scripts will be released upon publication.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, 2024.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160.

Yiming Du, Baojun Wang, Yifan Xiang, Zhaowei Wang, Wenyu Huang, Boyang Xue, Bin Liang, Xingshan Zeng, Fei Mi, Haoli Bai, Lifeng Shang, Jeff Z. Pan, Yuxin Jiang, and Kam-Fai Wong. Memory-T1: Reinforcement learning for temporal reasoning in multi-session agents. In *International Conference on Learning Representations (ICLR)*, 2026.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 13481–13544. PMLR, 2024.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. *arXiv preprint arXiv:2212.10496*, 2022.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10746–10761. Association for Computational Linguistics, 2025.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, and Yafeng Deng. EverMemOS: A self-organizing memory operating system for structured long-horizon reasoning. *arXiv preprint arXiv:2601.02163*, 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations (ICLR)*, 2025a.

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

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.naacl-long.389.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J. Taylor, and Dan Roth. Know me, respond to me: Benchmarking LLMs for dynamic user profiling and personalized responses at scale. In *The 2025 Conference on Language Modeling (COLM)*, 2025.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25961–25970. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1318.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations (ICLR)*, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhengren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofen Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. MemOS: A memory OS for AI system. *arXiv preprint arXiv:2507.03724*, 2025.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 32201–32223. PMLR, 2024.

Jiaqi Liu, Xinyu Ye, Peng Xia, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Evolvemem: Self-evolving memory architecture via autoresearch for LLM agents. *arXiv preprint arXiv:2605.13941*, 2026.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting for retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.747.

NevaMind-AI. memU: Your personal memory, across every agent. `https://github.com/NevaMind-AI/memU/tree/2c050bc9681a4c0aff1af211a000e73d14f33356`, 2025.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations (ICLR)*, 2026.

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

Zehua Pei, Hui-Ling Zhen, Shixiong Kai, Sinno Jialin Pan, Yunhe Wang, Mingxuan Yuan, and Bei Yu. SCOPE: Prompt evolution for enhancing agent effectiveness. *arXiv preprint arXiv:2512.15374*, 2025.

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

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations (ICLR)*, 2024.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2023.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Yu Wang and Xi Chen. MIRIX: Multi-agent memory system for LLM-based agents. *arXiv preprint arXiv:2507.07957*, 2025.

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. *arXiv preprint arXiv:2402.04624*, 2024.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending MemoryLLM with scalable long-term memory. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 63308–63323. PMLR, 2025.

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, and Christian Szegedy. Memorizing transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

Menglin Xia, Xuchao Zhang, Shantanu Dixit, Paramaguru Harimurugan, Rujia Wang, Victor Ruhle, Robert Sim, Chetan Bansal, and Saravan Rajmohan. Memora: A harmonic memory representation balancing abstraction and specificity. In *Proceedings of the 43rd International Conference on Machine Learning*. PMLR, 2026.

Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Wenlin Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, and Tong Xu. From single to multi-granularity: Toward long-term memory association and selection of conversational agents. In *International Conference on Learning Representations (ICLR)*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations (ICLR)*, 2024.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. Memagent: Reshaping long-context LLM with multi-conv RL-based memory agent. In *International Conference on Learning Representations (ICLR)*, 2026.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025. doi: 10.1038/s41586-025-08661-4.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations (ICLR)*, 2025.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026a.

Yuhan Zhang, Zhiyuan Guo, Ziheng Zeng, Wei Wang, Wentao Wu, and Lijie Xu. Mandol: An agglomerative agent memory system for long-term conversations. *arXiv preprint arXiv:2606.29778*, 2026b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, 2024.

Tianyu Zhao and Llion Jones. Fast-weight product key memory. *arXiv preprint arXiv:2601.00671*, 2026.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning LLM agents without fine-tuning LLMs. *arXiv preprint arXiv:2508.16153*, 2025.

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

A ALGORITHMS

This appendix gives the full pseudocode of memory generation, dynamic search, and the validation-driven evolutionary policy search. Notation follows the main text.

Algorithm 1 details memory generation, the Add operator of Eq. 1. It runs the four stages of Section 3.4 under the policy π.

Algorithm 1 Memory generation (the Add operator) under policy π.

Require: New messages m_t; store G_{t-1}; policy π
Ensure: Updated store G_t

$\{\epsilon_1, \ldots, \epsilon_n\} \leftarrow$ SEGMENTEPISODES(m_t) ▷ one sub-topic per episode
$G \leftarrow G_{t-1}$
for each episode ϵ_i **do**
  $S_i \leftarrow$ GENERATEMEMORIES$(\epsilon_i;\ \pi)$ ▷ entities and property values declared by π
  add one episodic entity to S_i ▷ fallback for context π does not capture
  $S_i \leftarrow$ FUSEENTITIES$(S_i,\ G)$ ▷ resolve equivalence; merge; revise timelines
  $S_i \leftarrow$ SYNTHESIZEHIGHERORDER$(S_i,\ G)$ ▷ properties declared by π, from the relevant history
  $G \leftarrow$ GRAPHMERGE$(G,\ S_i)$ ▷ fuse nodes, form edges
end for
return $G_t \leftarrow G$

Algorithm 2 details dynamic search, the Search operator of Eq. 1. The LLM controller iterates the plan-search-refine loop of Section 3.5.

Algorithm 2 Dynamic memory search (the Search operator) for query q.

Require: Query q; store G_t
Ensure: Entity-centered evidence view for q

$P \leftarrow$ PLAN(q) ▷ LLM controller: temporal scope, memory types, sub-queries
$C \leftarrow \emptyset$
while not SUFFICIENT(C) **do**
  for each sub-query q_j in P **do**
    $C_j^f \leftarrow$ FORWARD(q_j) ▷ $e \rightarrow p$; search-field augmentation
    $C_j^r \leftarrow$ REVERSE(q_j) ▷ $p \rightarrow e$
    $C \leftarrow C \cup C_j^f \cup C_j^r$ ▷ BM25 and dense rankings fused by RRF within each direction
  end for
  $C \leftarrow$ TEMPORALEXPAND(C) ▷ predecessor and successor versions
  $C \leftarrow$ RELATIONALEXPAND(C) ▷ associated entities
  $P \leftarrow$ REFINE$(P,\ C)$
end while
return ENTITYVIEW(C) ▷ deduplicated, entity-centered

Algorithm 3 details the validation-driven evolutionary loop of Section 3.6. The fitting set $\mathcal{D}$ is grouped into U clusters; the algorithm runs E epochs over them, $H = E \cdot U$ generations in total, and maintains a population of K policies, $\pi_{h,j}$ being the j-th individual at generation h. Fitness is Eq. 3 evaluated on the active cluster of the current generation, so every individual, elites included, is re-scored at every generation. The population size is $K = 10$; the algorithm runs $E = 2$ epochs over $U = 12$ clusters, i.e. $H = 24$ generations. The elite fraction is $\alpha = 0.2$, and the crossover probability is $p_{\text{cross}} = 0.5$. The size-regularizer weight is $\lambda = 0.1/M_u$, where M_u is the number of messages in the active cluster. Exploratory mutation prunes the bottom-20%-frequency elements with probability $p_{\text{prune}} = 0.2$.

Algorithm 3 Validation-driven evolutionary policy search.

Require: Fitting set $\mathcal{D}$; initial policy π_0; epochs E; population size K; elite fraction α; probabilities p_{prune} and p_{cross}
Ensure: Evolved policy π^*

$\{\mathcal{D}_1, \ldots, \mathcal{D}_U\} \leftarrow \textsc{PartitionClusters}(\mathcal{D})$ ▷ by context scope
Populate $\mathcal{P}_1$ with K random mutations of π_0
for $e = 1, \ldots, E$ **do**
  for $u = 1, \ldots, U$ **do** ▷ active cluster $\mathcal{D}_u$; generation $h = (e-1) \cdot U + u$
    for $j = 1, \ldots, K$ **do**
      $G_{h,j} \leftarrow A(\mathcal{C}_u; \pi_{h,j})$ ▷ fold Add over the cluster contexts on a sandboxed clean store
      $\hat{a}_q \leftarrow \text{LLM}(q \mid R(q; G_{h,j}))$ for each query q of $\mathcal{D}_u$ ▷ frozen answer model
      $f_{h,j} \leftarrow f(\pi_{h,j})$ of Eq. 3, evaluated on $\mathcal{D}_u$
    end for
    Sort $\mathcal{P}_h$ by fitness in descending order; keep the top α fraction as elites
    for each non-elite slot j of $\mathcal{P}_{h+1}$ **do**
      $p_1, p_2 \leftarrow \textsc{TournamentSelect}(\mathcal{P}_h)$
      $\pi \leftarrow \textsc{ErrorInformedMutate}(p_1)$ ▷ the LLM reads the evaluation errors of p_1, locates information gaps, and proposes new entity types, properties, or refined descriptions
      $\pi \leftarrow \textsc{ExploratoryMutate}(\pi)$ ▷ inject plausible types for the scenario; prune bottom-20%-frequency elements with probability p_{prune}
      $\pi_{h+1,j} \leftarrow \textsc{Crossover}(\pi, p_2)$ ▷ insert entities and properties of p_2 with probability p_{cross}
    end for
  end for
end for
return $\pi^* \leftarrow \arg\max_{\pi \in \mathcal{P}_{H+1}} f(\pi)$, with f of Eq. 3 evaluated on the full fitting set $\mathcal{D}$

B A STORE EXAMPLE

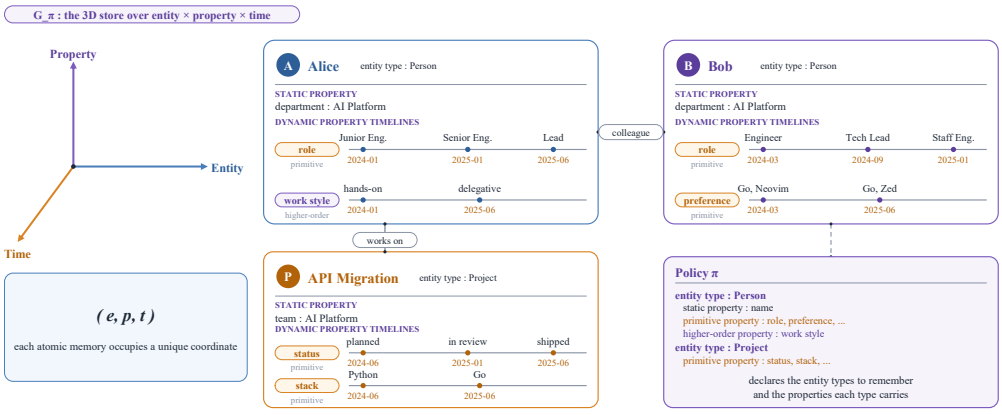


Figure 6: An example store G_π under a policy π. Static properties carry no timeline; primitive and higher-order properties keep time-ordered value sequences, and every concrete property value is an atomic memory at a coordinate (e, p, t). Relations link entities into a network.

Figure 6 shows a stored state of a small store G_π under one policy π. The policy declares the entity types to remember and the properties each type carries: persons with role and work-style properties, projects with status and stack, and so on. Every object that occurs in the messages is modeled as an entity; Alice, Bob, and the API Migration project are three instances. Static properties such as name carry no timeline. Primitive and higher-order properties both keep a time-ordered sequence of dated

values. Alice's role property, a primitive one, holds Junior Engineer, Senior Engineer, and Lead; the latest value never overwrites the history. Her work-style property, a higher-order one, is synthesized from the history and also keeps dated values. Each concrete property value is an atomic memory at a coordinate (e, p, t). Relations among entities form edges: Alice is a colleague of Bob and works on the API Migration project.

C POLICY MANIFESTS

This appendix lists the four policy configurations compared in Section 4.3 and Figure 5: the initial policy π_0 (the raw baseline), the human-designed general policy (the pre-defined policy of Table 1), the human-designed PersonaMem policy (the pre-defined policy of Table 2), and the evolved policy (the output of Section 3.6). Each manifest is shown in JSON. The episodic entity fallback (Section 3.4) is disabled in the listed manifests. A dynamic property of order 1 is primitive and order 2 is higher-order (Section 3.3). Long descriptions are truncated to their first sentence. The complete manifests are released with the code.

C.1 INITIAL POLICY π_0

```
[
  {
    "entity_type": "user",
    "entity_description": "The primary user/speaker in the
        conversation.",
    "static_property": {"name": "User name or identifier"},
    "dynamic_property": {
      "food_preference": {"type": "string", "order": 1, "desc":
          "[Analytical Summary] The user's dietary and dining
          preferences: foods, cuisines, flavors, restaurants, cooking
          or eating habits the user clearly likes, dislikes, or
          prefers."},
      "travel_preference": {"type": "string", "order": 1, "desc":
          "[Analytical Summary] The user's travel preferences:
          destinations, travel styles, trip activities, and travel
          planning attitudes the user clearly likes, dislikes, or
          prefers."}
    }
  }
]
```

C.2 HUMAN-DESIGNED GENERAL POLICY

```
[
  {
    "entity_type": "person",
    "entity_description": "All persons appearing in the dialogue,
        including the speaker and mentioned people.",
    "static_property": {"name": "Person name, must be a specific name",
        "birth_info": "Birth information, such as birth date or age
        range"},
    "dynamic_property": {
      "location_event": {"type": "string", "desc": "Location change
          event, including time, old location, new location."},
      "position_event": {"type": "string", "desc": "Profession change
          event, including time, position."},
      "health_event": {"type": "string", "desc": "Health change event,
          including time, health condition, injuries, recovery."},
      "skills_event": {"type": "string", "desc": "Skill acquisition
          event, including time, skill name."},
      "mood_event": {"type": "string", "desc": "Mood change event,
          including time, emotion, reason."},
```

```
    "habit_event": {"type": "string", "desc": "Habit change event,
        including time, old habit, new habit."},
    "experience": {"type": "string", "desc": "Personal experience,
        direct text description not list."},
    "plan_event": {"type": "string", "desc": "Future plan or
        intention, including time, goal, target."},
    "relationship_status": {"type": "string", "desc":
        "Relationship/marital status, including time and status."},
    "education_field": {"type": "string", "desc": "Education field or
        major of study."},
    "education_degree": {"type": "string", "desc": "Educational
        degree or certification."},
    "career_interest": {"type": "string", "desc": "Career interests
        or aspirations."},
    "identity": {"type": "string", "desc": "Personal identity,
        including gender identity, cultural identity, community
        membership."},
    "family_plan": {"type": "string", "desc": "Family-related plans
        such as adoption, having children, marriage."},
    "business_event": {"type": "string", "desc": "Business-related
        events such as starting a business, launching a product,
        opening a store."},
    "hobby_activity": {"type": "string", "desc": "Hobby and
        recreational activities, including hobbies, interests, and
        regular activities."},
    "achievement_event": {"type": "string", "desc": "Achievements and
        awards, including competition wins, certifications,
        recognitions."},
    "reading_activity": {"type": "string", "desc": "Books read or
        reading activities."},
    "travel_event": {"type": "string", "desc": "Travel events and
        trips, including specific destinations, companions, and
        activities."},
    "social_activity": {"type": "string", "desc": "Social activities
        and networking events."},
    "preference": {"type": "string", "desc": "Personal preferences,
        likes, dislikes, favorites, opinions, and impressions of
        other people or characters."},
    "religious_belief": {"type": "string", "desc": "Religious or
        spiritual beliefs and practices."},
    "allergy_condition": {"type": "string", "desc": "Allergies,
        dietary restrictions, or health conditions that constrain
        daily life."},
    "recommendation_given": {"type": "string", "desc":
        "Recommendations or suggestions given to or received from
        others."},
    "creative_work": {"type": "string", "desc": "Creative works or
        artistic projects, including writing, painting, music, video
        production, screenplays, scripts."},
    "training_event": {"type": "string", "desc": "Training, courses,
        or structured learning activities attended."},
    "volunteer_event": {"type": "string", "desc": "Volunteering
        activities, community service, or charitable participation."},
    "pet_info": {"type": "string", "desc": "Brief record of pet
        ownership relationship."},
    "collection_hobby": {"type": "string", "desc": "Items collected,
        collectibles, or collecting hobbies."},
    "language_skill": {"type": "string", "desc": "Languages spoken,
        studied, or being learned."},
    "cooking_activity": {"type": "string", "desc": "Cooking
        activities, recipes, food preparation, and culinary
        interests."},
    "gaming_event": {"type": "string", "desc": "Gaming activities,
        tournaments, competitions, and gaming achievements."},
```

```
    "writing_project": {"type": "string", "desc": "Writing projects
        including screenplays, books, blogs, stories."},
    "financial_event": {"type": "string", "desc": "Financial events
        including donations, purchases, investments, financial status
        changes."},
    "sports_event": {"type": "string", "desc": "Sports activities,
        competitions, and athletic achievements."},
    "room_decoration": {"type": "string", "desc": "Room decorations,
        home setup, furniture, and living space descriptions."},
    "personal_item": {"type": "string", "desc": "Important personal
        items, keepsakes, tools, and meaningful objects."},
    "motivation_method": {"type": "string", "desc": "Methods used for
        staying motivated, focused, or productive."},
    "nickname": {"type": "string", "desc": "Nicknames, pet names, or
        alternate names used by or for a person."},
    "media_consumption": {"type": "string", "desc": "Movies, TV
        shows, music, podcasts watched or listened to."},
    "endorsement_deal": {"type": "string", "desc": "Endorsement
        deals, sponsorships, brand partnerships."}
  }
},
{
  "entity_type": "organization",
  "entity_description": "Organizations or institutions, such as
      companies, teams, clubs, etc.",
  "static_property": {"name": "Organization name", "legal_form":
      "Legal form, such as Limited, Association, etc."},
  "dynamic_property": {
    "membership_event": {"type": "string", "desc": "Membership change
        event, including time, member name, role."},
    "operational_status_event": {"type": "string", "desc":
        "Operational status change event, including time, status."},
    "location_info": {"type": "string", "desc": "Geographic location
        or address of the organization."},
    "service_info": {"type": "string", "desc": "Services, programs,
        or activities offered by the organization."},
    "partnership_event": {"type": "string", "desc": "Partnerships,
        collaborations, or deals with people or other organizations."}
  }
},
{
  "entity_type": "item",
  "entity_description": "Significant physical items, gifts,
      keepsakes, artworks, or objects that carry personal meaning or
      are referenced multiple times.",
  "static_property": {"name": "Item name or descriptive identifier",
      "item_type": "Category of the item (e.g., stuffed animal,
      painting, book, decoration, equipment)"},
  "dynamic_property": {
    "ownership_event": {"type": "string", "desc": "Ownership or
        acquisition event."},
    "usage_context": {"type": "string", "desc": "How the item is used
        or its significance."},
    "location_info": {"type": "string", "desc": "Where the item is
        located or displayed."}
  }
},
{
  "entity_type": "animal",
  "entity_description": "Named pets or animals that are meaningfully
      discussed in the dialogue, including pets owned by speakers,
      stray animals they interact with, or animals central to a
      story.",
```

```
    "static_property": {"name": "Animal name or descriptive
        identifier", "species": "Species and breed (e.g., golden
        retriever, tabby cat, parrot)", "owner": "Owner's name"},
    "dynamic_property": {
      "adoption_event": {"type": "string", "desc": "Adoption or
          acquisition event."},
      "health_event": {"type": "string", "desc": "Health-related events
          including vet visits, illnesses, injuries, vaccinations."},
      "training_event": {"type": "string", "desc": "Training
          activities, classes, or behavioral milestones."},
      "activity_event": {"type": "string", "desc": "Activities,
          outings, or notable behaviors."},
      "characteristic": {"type": "string", "desc": "Personality traits,
          habits, preferences, or physical characteristics."}
    }
  },
  {
    "entity_type": "place",
    "entity_description": "Specific named locations, venues, or
        geographic places that are meaningfully discussed in the
        dialogue.",
    "static_property": {"name": "Place name or geographic identifier",
        "place_type": "Category (e.g., park, restaurant, city, hiking
        trail, national park, shelter, school)"},
    "dynamic_property": {
      "visit_event": {"type": "string", "desc": "Visit or activity at
          the place."},
      "notable_feature": {"type": "string", "desc": "Notable features,
          characteristics, or facts about the place."}
    }
  }
]
```

C.3 HUMAN-DESIGNED PERSONAMEM POLICY

```
[
  {
    "entity_type": "user",
    "entity_description": "The primary user/speaker in the
        conversation.",
    "static_property": {"name": "User name or identifier"},
    "dynamic_property": {
      "location_event": {"type": "string", "order": 1, "desc":
          "[Factual Record] Location change event, including time, old
          location, new location."},
      "position_event": {"type": "string", "order": 1, "desc":
          "[Factual Record] Profession change event, including time,
          position, company."},
      "health_event": {"type": "string", "order": 1, "desc": "[Factual
          Record] Health change event, including time, health
          condition, injuries, recovery."},
      "skills_event": {"type": "string", "order": 1, "desc": "[Factual
          Record] Skill acquisition event, including time, skill
          name."},
      "mood_event": {"type": "string", "order": 1, "desc": "[Analytical
          Summary] Emotional state or mood."},
      "habit_event": {"type": "string", "order": 1, "desc":
          "[Analytical Summary] Behavioral pattern change."},
      "experience": {"type": "string", "order": 1, "desc": "[Factual
          Record] Personal experience, direct text description."},
      "plan_event": {"type": "string", "order": 1, "desc": "[Factual
          Record] Future plan or intention, including time, goal,
          target."},
```

```
    "relationship_status": {"type": "string", "order": 1, "desc":
        "[Factual Record] Relationship/marital status, including time
        and status."},
    "education_field": {"type": "string", "order": 1, "desc":
        "[Factual Record] Education field or major of study."},
    "education_degree": {"type": "string", "order": 1, "desc":
        "[Factual Record] Educational degree or certification."},
    "career_interest": {"type": "string", "order": 1, "desc":
        "[Analytical Summary] Career interests or aspirations."},
    "identity": {"type": "string", "order": 1, "desc": "[Factual
        Record] Personal identity, including gender identity,
        cultural identity, community membership."},
    "family_plan": {"type": "string", "order": 1, "desc": "[Factual
        Record] Family-related plans such as adoption, having
        children, marriage."},
    "business_event": {"type": "string", "order": 1, "desc":
        "[Factual Record] Business-related events such as starting a
        business, launching a product, opening a store."},
    "hobby_activity": {"type": "string", "order": 1, "desc":
        "[Factual Record] Hobby and recreational activities,
        including hobbies, interests, and regular activities."},
    "achievement_event": {"type": "string", "order": 1, "desc":
        "[Factual Record] Achievements and awards, including
        competition wins, certifications, recognitions."},
    "reading_activity": {"type": "string", "order": 1, "desc":
        "[Factual Record] Books read or reading activities."},
    "travel_event": {"type": "string", "order": 1, "desc": "[Factual
        Record] Travel events and trips, including specific
        destinations, companions, and activities."},
    "social_activity": {"type": "string", "order": 1, "desc":
        "[Factual Record] Social activities and networking events."},
    "preference": {"type": "string", "order": 1, "desc": "[Analytical
        Summary] Personal preferences, likes, dislikes, favorites."},
    "preference_evolution": {"type": "string", "order": 1, "desc":
        "[Analytical Summary] Preference change over time."},
    "opinion": {"type": "string", "order": 1, "desc": "[Analytical
        Summary] User's definitive opinion or view on a specific
        topic."},
    "attitude_change": {"type": "string", "order": 1, "desc":
        "[Analytical Summary] Attitude or stance change over time."},
    "financial_status": {"type": "string", "order": 1, "desc":
        "[Analytical Summary] Financial situation, strategy, or
        status change."},
    "advice_given": {"type": "string", "order": 1, "desc":
        "[Analytical Summary] Key advice or recommendations
        received."},
    "default_property": {"type": "string", "order": 1, "desc":
        "Catch-all property for events, facts, or information that do
        not fit into any other defined property category."},
    "personality_trait": {"type": "string", "order": 2, "desc":
        "[Higher Order] Personality characteristic inferred from
        multiple memories."},
    "behavioral_pattern": {"type": "string", "order": 2, "desc":
        "[Higher Order] Recurring behavioral pattern observed across
        multiple events."},
    "value_orientation": {"type": "string", "order": 2, "desc":
        "[Higher Order] Core value or life priority inferred from
        choices and opinions."},
    "decision_style": {"type": "string", "order": 2, "desc": "[Higher
        Order] How the user typically approaches decisions, inferred
        from multiple decision events."},
    "interpersonal_style": {"type": "string", "order": 2, "desc":
        "[Higher Order] How the user interacts with others,
        communication patterns, social tendencies."},
```

```
    "preference_summary": {"type": "string", "order": 2, "desc":
        "[Higher Order] Current panoramic summary of the user's
        active preferences across domains."},
    "interest_domain": {"type": "string", "order": 2, "desc":
        "[Higher Order] Core interest area or passion domain
        synthesized from ALL available activities, hobbies, and
        preferences - not just the latest episode."},
    "change_pattern": {"type": "string", "order": 2, "desc": "[Higher
        Order] Meta-pattern of how the user's preferences or
        attitudes typically evolve over time."}
   }
  },
  {
   "entity_type": "fact",
   "entity_description": "Any factual information mentioned in the
       dialogue, including meetings, activities, events, important
       information, agreements, etc.",
   "static_property": {"subject": "Fact topic or core content",
       "category": "Category such as agreement, password, address,
       plan, knowledge, event, meeting"},
   "dynamic_property": {
    "fact_event": {"type": "string", "order": 1, "desc": "Fact event,
       including time, participants, content."},
    "fact_update": {"type": "string", "order": 1, "desc": "Fact
       update or evolution over time."}
   }
  },
  {
   "entity_type": "organization",
   "entity_description": "Organizations or institutions, such as
       companies, teams, clubs, etc.",
   "static_property": {"name": "Organization name", "legal_form":
       "Legal form, such as Limited, Association, etc."},
   "dynamic_property": {
    "membership_event": {"type": "string", "order": 1, "desc":
       "Membership change event, including time, member name,
       role."},
    "operational_status_event": {"type": "string", "order": 1,
       "desc": "Operational status change event, including time,
       status."},
    "location_info": {"type": "string", "order": 1, "desc":
       "Geographic location or address of the organization."},
    "service_info": {"type": "string", "order": 1, "desc": "Services,
       programs, or activities offered by the organization."},
    "partnership_event": {"type": "string", "order": 1, "desc":
       "Partnerships, collaborations, or deals with people or other
       organizations."},
    "default_property": {"type": "string", "order": 1, "desc":
       "Catch-all property for organization-related events or facts
       that do not fit into any other defined property category."}
   }
  }
]
```

C.4 EVOLVED POLICY

```
[
  {
   "entity_type": "user",
   "entity_description": "The primary user/speaker in the
       conversation.",
   "static_property": {"name": "User name or identifier"},
   "dynamic_property": {
```

```
    "finance_event": {"type": "string", "order": 1, "desc": "Record
        one concise, specific episode in which the user directly
        takes, plans, pauses, changes, evaluates, or makes a decision
        about their own finances."},
    "decision_making_style": {"type": "string", "order": 2, "desc":
        "Capture durable patterns in how the user makes decisions
        across domains, such as being deliberate or spontaneous,
        evidence-seeking or intuition-led, risk-averse or
        risk-tolerant, preference-sensitive, consensus-oriented, or
        inclined to test options incrementally."},
    "family_relationship_event": {"type": "string", "order": 1,
        "desc": "Record one distinct, explicitly stated episode or
        lifecycle change involving the user and a family member,
        household member, relative, or shared community group when
        the activity is undertaken with, for, or to strengthen a
        family or close relationship."},
    "family_relationships_feature": {"type": "string", "order": 2,
        "desc": "Capture durable, synthesized patterns in how the
        user builds and maintains relationships with family members
        through shared activities, projects, traditions, meals,
        outings, or other bonding formats."},
    "health_wellness_event": {"type": "string", "order": 1, "desc":
        "Record explicit, concrete episodes concerning the user's
        physical health, mental health, medical care, treatment,
        recovery, injury, sleep, nutrition, exercise, or other
        personal wellness activity."},
    "financial_management_orientation": {"type": "string", "order":
        2, "desc": "Capture durable, synthesized patterns in how the
        user approaches money management and finance-related learning
        across multiple conversations or explicitly comparative
        reflections."},
    "personal_engagement_event": {"type": "string", "order": 2,
        "desc": "Record explicit concrete episodes and well-supported
        higher-order patterns in how the user actively engages with
        creative, cultural, educational, social, community, hobby,
        discussion, and wellness/self-management interests."},
    "work_career_event": {"type": "string", "order": 1, "desc":
        "Record one concrete, specific episode in the user's
        employment or career development, whether completed, ongoing,
        or explicitly planned."},
    "travel_event": {"type": "string", "order": 1, "desc": "Record
        one concise, specific episode involving the user's travel or
        trip planning."}
  }
 }
]
```

The evolved manifest contains no catch-all property: every property is scoped to one facet of the user. This is the structural counterpart of the memory-cost result in Figure 5, where the evolved policy keeps the smallest store among the three designed policies.