

Mnemis: Dual-Route Retrieval on Hierarchical Graphs for Long-Term LLM Memory

Anonymous ACL submission

Abstract

AI Memory, specifically how models organize and retrieve historical messages, becomes increasingly valuable to Large Language Models (LLMs), yet existing methods (RAG and Graph-RAG) primarily retrieve memory through similarity-based mechanisms. While efficient, such System-1-style retrieval struggles with scenarios that require global reasoning or comprehensive coverage of all relevant information. In this work, We propose Mnemis, a novel memory framework that integrates System-1 similarity search with a complementary System-2 mechanism, termed Global Selection. Mnemis organizes memory into a base graph for similarity retrieval and a hierarchical graph that enables top-down, deliberate traversal over semantic hierarchies. By combining the complementary strength from both retrieval routes, Mnemis retrieves memory items that are both semantically and structurally relevant. Mnemis achieves state-of-the-art performance across all compared methods on long-term memory benchmarks, scoring 93.9 on LoCoMo and 91.6 on LongMemEval-S using GPT-4.1-mini.

1 Introduction

With the rapid advancement of Large Language Models (LLMs), there is a growing trend to integrate memory mechanisms to support long-term interactions (Lewis et al., 2020; Ouyang et al., 2025; Behrouz et al., 2024). As LLMs shift from text generators to persistent interactive agents, the ability to organize and retrieve past interactions becomes increasingly valuable. The prevailing research paradigm is based on retrieval-augmented generation (RAG). Inspired by human episodic memory (Tulving et al., 1972), these methods (e.g. SeCom (Pan et al., 2025), Memory-R1 (Yan et al., 2025)) explicitly store historical messages (i.e., *Episodes*) and retrieve only the most relevant pieces (Arslan

et al., 2024; Lewis et al., 2020). This design alleviates the computational and latency issues of long-context models and keeps the input compact and focused. However, its effectiveness critically depends on retrieval quality.

Recent work on graph-based RAG (Graph-RAG) extends RAG by incorporating concepts from semantic memory (Tulving et al., 1972). Graph-RAG (e.g. GraphRAG (Edge et al., 2024), Nemori (Nan et al., 2025), Mem0 (Chhikara et al., 2025), and Zep (Rasmussen et al., 2025)) extracts memory segments, e.g. *Entities* (key figures, objects, or concepts) and *Edges* (events or relationships connecting Entities) and organizes memory into a structured graph. These methods highlight essential information and enable more effective and semantically meaningful retrieval.

Although Graph-RAG-based methods mark an important step toward structured memory, their retrieval remains largely similarity-driven, selecting Episodes, Entities, or Edges via text matching (BM25) or embedding similarity (cosine). This approach is fast and effective, and resembles the System-1 process in dual-process theory (Kahne-man, 2011), but becomes limited when queries require global reasoning or comprehensive coverage of all relevant information. Although recent research has explored iterative generation of sub-queries to mitigate this issue, such methods still fall short for questions that require a broader perspective (Wang et al., 2025; Jin et al., 2025). For example, consider the query "Which cities did Dave travel to in 2023?" from LoCoMo Benchmark (Maharana et al., 2024), as shown in fig. 1. The mention "attended a conference in Detroit." is buried in a long message and has only a weak semantic relation to the user query. Moreover, generating effective sub-queries is challenging as the model lacks a global view of the memory to determine how the original query should be meaningfully expanded.

Recalling how human approach such questions,

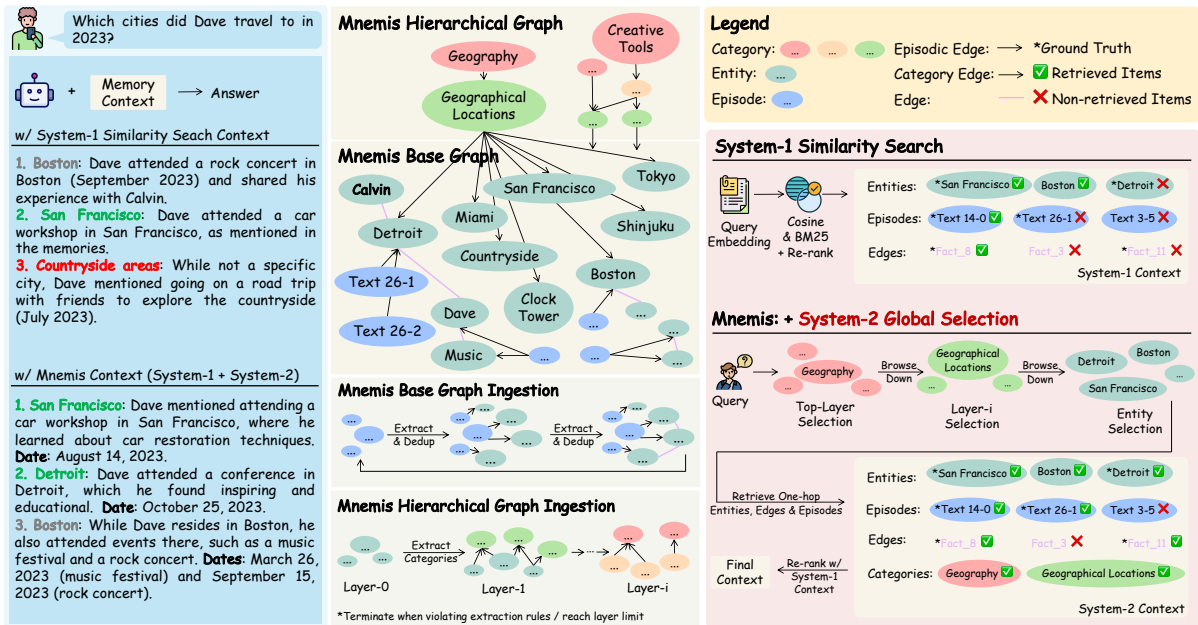


Figure 1: Framework of Mnemis together with the workflow of base graph ingestion, hierarchical graph ingestion and search. Left is a real case from LoCoMo.

it can be naturally addressed using a semantic hierarchy. We can begin with a high-level concept (*e.g.* city), enumerate all the cities we have visited and verify them one-by-one. This kind of solution operates over a global view of memory and naturally avoids the need for sub-query generation, reflecting a structured process characteristic of System-2 reasoning (Kahneman, 2011).

Inspired by this observation, we propose an analogous mechanism, called *Global Selection*, which constructs a hierarchical graph that provides a complete, global, and structured view of the entire memory, mimicking human semantic hierarchies. It allows models to perform top-down, deliberate memory scanning within it. In this example, Global Selection can start from the top layer and follow the path "Geography" → "Geographical Locations" → "Detroit" to retrieve the relevant information.

In practice, real-world queries often benefit from combining both the System-1 and System-2 processes, as they operate through different retrieval patterns. Motivated by this, we present Mnemis, a novel and effective framework to organize and retrieve AI memory. Mnemis comprises two storage components: a base graph and a hierarchical graph, and two corresponding retrieval routes: System-1 similarity search and System-2 global selection. The base graph, similar to prior Graph-RAG designs, extracts Entities and Edges from history texts (Episodes) to support similarity-based retrieval. We refine the extraction pipeline to increase extraction fields and improve extraction quality. In contrast,

the hierarchical graph prompts LLMs to categorize Entities into higher-level Categories through bottom-up. This process follows three key principles: (1) Minimum Concept Abstraction: each Category should faithfully capture the shared features of its child nodes. It should be specific enough to be informative, yet sufficiently general to support abstraction; (2) Many-to-Many Mapping: one child node can be assigned to multiple Categories to represent its different semantic facets; and (3) Compression Efficiency Constraint: one Category must contain at least n children and higher layers must contain no more Categories than lower layers (applied from layer 2 onward).

When a query arrives, the similarity search route conducts a semantic search based on embeddings and text similarity, while the global search performs a top-down selection through the hierarchical graph, layer by layer. Down to the lowest level, the LLM first selects all relevant entities and then retrieves all edges, entities and episodes connected to them. These two routes capture complementary signals: System-1 provides fine-grained semantic similarity evidence, while System-2 retrieves structurally relevant items that may be semantically distant yet relationally important. By combining and re-ranking the union of both routes, Mnemis achieves SOTA performance across all compared methods on long-term memory benchmarks, scoring 93.9 on LoCoMo and 91.6 on LongMemEval-S using GPT-4.1-mini. Our contributions can be summarized as below:

- We introduce Mnemis, a novel framework that integrates System-1 similarity search with System-2 global selection to perform both semantic retrieval and deliberate, top-down reasoning over memory;
- We improve the base graph extraction and construct a hierarchical graph for global selection, guided by Minimum Concept Abstraction, Many-to-Many Mapping, and Compression Efficiency Constraint;
- We perform comprehensive experiments to demonstrate the effectiveness of Mnemis. Mnemis achieves SOTA performance on long-term memory benchmarks, scoring 93.9 on LoCoMo and 91.6 on LongMemEval-S using GPT-4.1-mini.

2 Mnemis Methodology

To achieve effective memory organization, Mnemis constructs two major components: a base graph and a hierarchical graph and two key memory retrieval mechanisms: System-1 Similarity Search and System-2 Global Selection. We implement Mnemis based on Graphiti¹.

2.1 Base Graph

The base graph stores historical messages and captures detailed information, enabling Mnemis to perform System-1 Similarity Search. It consists of four components: Episodes, Entities, Edges and Episodic Edges.

Episodes. Each episode is a piece of raw historical text. It is encoded into an `episode_embedding` for similarity-based retrieval. Its timestamp is recorded at `valid_at`.

Entities. An entity is any concrete person, organization, place, object, event, or well-defined concept. Each entity includes `name`, `summary`, `tag`, and `episode_idx`. The `summary` provides a concise contextual description, the `tag` specifies its type or role, and `episode_idx` tracks the episodes it appears. We encode `name` and `summary` into corresponding embeddings for flexible search.

Edges. An edge is a verifiable statement describing a meaningful relationship, action, or state involving one or more specified entities within a defined temporal or contextual scope. Each edge connects two entities through a `fact`, which is encoded as a `fact_embedding`. Additionally,

`valid_at` and `invalid_at` specify the time span during which the edge is considered valid.

Episodic Edges. An episodic edge links entities to all episodes where they appear. It is utilized during global search to retrieve all episodes associated with selected entities.

The ingestion of the base graph is conducted incrementally: new inputs are first formatted into Episodes. Based on their timestamps, recent Episodes will be retrieved to provide additional context. During extraction, the LLM first identifies entity names from both the current and recent Episodes, followed by a reflection process to capture omitted entity names. These names are then de-duplicated against existing entities in memory, using a combination of full-text search and similarity search over the `name_embedding`. After de-duplication, each entity’s `summary`, `tag`, and `episode_idx` are extracted according to the episode context. Subsequently, Edges are extracted using both Episodes and Entities as contextual inputs, followed by reflection and de-duplication steps analogous to those used in entity extraction.

2.2 Hierarchical Graph

The hierarchical graph abstracts Entities (layer 0) into multi-level Categories, enabling the LLM to perform System-2 Global Selection. The structure consists of two components, as shown in fig. 2.

Category Nodes (Categories). A category represents an abstract, high-level concept derived from lower-layer categories (or entities at layer 0). It shares the same core fields as an Entity, with an additional attribute layer indicating its position within the hierarchical graph.

Category Edges. A category edge links a higher-layer category to its child nodes (either lower-layer categories or entities). These edges define the hierarchical organization of the graph and support the top-down traversal process in global selection.

The ingestion of hierarchical graph is governed by three key design principles:

Minimum Concept Abstraction. While categories are intended to capture the shared semantics of their child nodes, we explicitly prompt the LLM to perform *minimal abstraction*. The resulting category should remain sufficiently specific to preserve informative detail, leaving room for broader generalizations at higher layers.

Many-to-Many Mapping. Unlike conventional tree-structured hierarchies, Mnemis permits lower-layer nodes to belong to multiple higher-layer cat-

¹[urlhttps://github.com/getzep/graphiti](https://github.com/getzep/graphiti)

egories. This design allows the hierarchy to represent different semantic facets of each node, enabling retrieval from multiple perspectives depending on the query.

Compression Efficiency Constraint. To ensure the efficiency of System-2 Global Selection, the hierarchy is regulated by two complementary mechanisms: (1) the *compression ratio* n and (2) the *node count reduction rule*, which takes effect from layer 2 onward.

The compression ratio constrains the hierarchy at the category level. Each category must contain at least n child nodes. An exception is made for nodes that cannot be naturally merged with others; such nodes are directly promoted to the next layer as standalone categories, encouraging meaningful aggregation while preventing overly fine-grained or trivial categories.

The node count reduction rule, in contrast, constrains the hierarchy at the layer level: each upper layer must contain no more nodes than the layer beneath it, ensuring progressive abstraction across layers. If this rule is violated, *e.g.*, when multiple nodes are promoted directly without merging and the result layer is oversized, the ingestion process is terminated to maintain hierarchical balance.

Guided by the principles above, the hierarchical graph is constructed layer by layer. At layer i , all nodes from layer $i - 1$ are first retrieved. Category names are then generated, and lower-layer nodes are assigned to these categories using their names and tags as contextual information. The construction process terminates when either the compression efficiency constraints are violated or the maximum layer limit is reached.

When the base graph is updated, the hierarchical graph should be updated accordingly. Currently, we periodically rebuild the hierarchical graph for simplicity and leave optimization for future work.

2.3 Memory Retrieval Mechanisms

Mnemis contains two major memory retrieval routes: System-1 Similarity Search and System-2 Global Selection. Given a user query, Mnemis retrieves Episodes, Entities and Edges and prompts LLM to get the final answer.

System-1 Similarity Search. This route retrieves the top- k Episodes, Entities, and Edges, providing fast and effective retrieval based on semantic similarity. It operates through two complementary methods: embedding search, which retrieves relevant items by computing cosine sim-

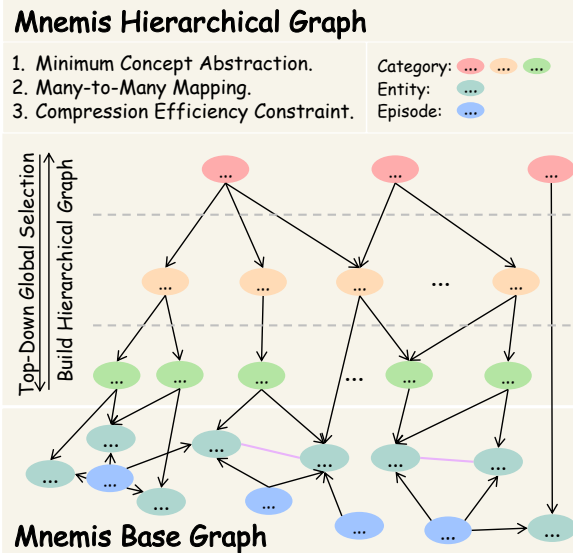


Figure 2: Mnemis Hierarchical Graph Overview.

ilarity between the query embedding and the corresponding embeddings (summary_embedding for Entities, fact_embedding for Edges, and episode_embedding for Episodes); and full-text search, which retrieves relevant components using BM25 over textual content (content for Episodes, name and summary for Entities, and fact for Edges). These two results are then merged and re-ranked using reciprocal rank fusion (RRF) (Cormack et al., 2009), which computes a fusion score by summing the reciprocals of each candidate’s ranks and orders candidates in descending score. Episodes, Entities, and Edges are re-ranked separately and then truncated to top- k according to the predefined search budget.

System-2 Global Selection. This route enables deliberate, top-down exploration of memory through the hierarchical graph. Because the process is primarily structure-driven and the selection at each layer is fully determined by the LLM, no strict top- k constraint is applied. Starting from the top layer, the LLM uses category names and tags to select relevant Categories based on the user query and progressively browses down the hierarchy. At the lowest level, all relevant entities are first selected. Mnemis then retrieves all episodes and edges directly connected to these entities, along with the entities linked through those edges.

Re-ranking. After executing both retrieval routes, we apply a re-ranking model to leverage their complementary strengths.² Episodes, Entities

²As System-2 produces unordered results, we cannot directly apply an RRF re-ranker as in System-1.

Table 1: Detailed performance (LLM-as-a-Judge score) on LoCoMo by question type. Following the common practice, Category 5 (Adversarial) is excluded from the results.

LLM	Methods	Multi-Hop	Temporal	Open-Domain	Single-Hop	Overall	
#Questions		282	321	96	841	1540	
GPT-4o-mini	Full Context	66.8	56.2	48.6	83.0	72.3	
	RAG	59.9	62.9	63.5	73.5	68.2	
	LangMem	52.4	24.9	47.6	61.4	51.3	
	MemOS	64.3	73.2	55.2	78.4	73.3	
	Mem0	60.3	50.4	40.6	68.1	61.3	
	Zep	50.5	58.9	39.6	63.2	58.5	
	Nemori	65.3	71.0	44.8	82.1	74.4	
	EMem-G	74.7	76.0	57.3	82.3	78.0	
	Mnemis	89.7	77.6	79.2	95.7	89.8	
GPT-4.1-mini	Full Context	77.2	74.2	56.6	86.9	80.6	
	RAG	64.9	76.6	67.7	76.5	73.8	
	LangMem	71.0	50.8	59.0	84.5	73.4	
	Mem0	68.2	56.9	47.9	71.4	66.3	
	Zep	53.7	60.2	43.8	66.9	61.6	
	Nemori	75.1	77.6	51.0	84.9	79.5	
	PREMem	61.0	74.8	46.9	66.2	65.8	
	EverMemOS	91.1	89.7	70.8	96.1	92.3	
	EMem-G	79.6	80.8	71.7	90.5	85.3	
		Mnemis	91.8	90.3	82.3	96.2	93.3
		Mnemis ($k=30$)	92.9	90.7	79.2	97.1	93.9

(Categories), and Edges are re-ranked separately. They are then reformatted into a unified memory context and provided to the answer model, together with the user query, to generate the final response.

3 Experiments

3.1 Experiment Setups

Datasets. We evaluate Mnemis on two well-known AI memory benchmarks: LoCoMo (Maharana et al., 2024) and LongMemEval-S (Wu et al., 2024). LoCoMo consists of long-term conversations from 10 users, with each user contributing approximately 600 turns across 32 sessions, totaling around 16K tokens on average. The dataset contains roughly 2,000 questions spanning five diverse categories: Single-Hop, Multi-Hop, Temporal, Open-Domain, and Adversarial. LongMemEval-S comprises 500 sessions, with each session containing one question and roughly 115K tokens, designed to evaluate five core memory abilities: information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention.

Baselines. We compare Mnemis against the following baselines: LangMem³, MemOS (Li et al., 2025), Mem0 (Chhikara et al., 2025), Zep (Rasmussen et al., 2025), Nemori (Nan et al., 2025), PreMem (Kim et al., 2025), EverMemOS⁴, EMem-G

(Zhou, 2025) using GPT-4o-mini or GPT-4.1-mini as the backend model for memory building and question answering. We directly use their reported performance. In addition, we include two supplementary baselines: Full Context, which feeds the entire conversation history to the model, and RAG, which retrieves only episodes while keeping all other settings identical to Mnemis. We also identified several other comparable baselines; however, due to missing details such as the backbone model and hyperparameter settings, we report their results only in appendix C.

Hyperparameters. Following Nemori (Nan et al., 2025), we limit the number of retrieved episodes in the answer prompt to top- $k = 10$, while entities (including categories) and edges are limited to top- $2k = 20$. We use Qwen3-Embedding-0.6B as the embedding model, with the embedding dimension fixed at 128 due to storage constraints. The re-ranker model used in the main experiments is Qwen3-Reranker-8B (Zhang et al., 2025). We use neo4j⁵ as the backend database. Across all experiments, the grader model is consistently GPT-4.1-mini to ensure accurate scoring.

Metrics. We employ LLM-as-a-Judge score (0/1) for evaluation and adopt the official judge prompt for each dataset. Following previous methods, Category 5 is excluded from LoCoMo’s score.

³<https://github.com/langchain-ai/langmem>

⁴<https://github.com/EverMind-AI/EverMemOS/>

⁵<https://neo4j.com/>

Table 2: Detailed performance (LLM-as-a-Judge score) on LongMemEval-S, categorized by question type: single-session-user (SSU), multi-session (MS), single-session-preference (SSP), temporal reasoning (TR), knowledge update (KU), and single-session-assistant (SSA).

LLM	Methods	SSU	MS	SSP	TR	KU	SSA	Overall
#Questions		70	133	30	133	78	56	500
GPT-4o-mini	Full Context	78.6	38.3	6.7	42.1	78.2	89.3	55.0
	RAG	88.6	47.4	<u>70.0</u>	63.2	70.5	91.1	67.2
	Mem0	91.4	66.2	34.0	63.9	74.4	<u>96.4</u>	71.1
	Zep	<u>92.9</u>	47.4	53.3	54.1	74.4	75.0	63.2
	Nemori	88.6	51.1	46.7	61.7	61.5	83.9	64.2
	EMem-G	87.0	<u>73.6</u>	32.2	<u>74.8</u>	94.4	87.5	<u>77.9</u>
	Mnemis	97.1	76.7	90.0	83.5	<u>92.3</u>	100.0	87.2
GPT-4.1-mini	Full Context	85.7	51.1	16.7	60.2	76.9	98.2	65.6
	RAG	82.9	54.9	86.7	67.7	80.8	94.6	72.6
	PREMem	92.9	57.1	36.7	59.4	84.6	12.5	60.8
	Mem0	94.3	66.9	86.7	75.9	87.2	<u>96.4</u>	80.8
	Nemori	90.0	55.6	86.7	72.2	79.5	92.9	74.6
	EverMemOS	100.0	78.5	<u>96.7</u>	71.2	87.2	78.6	82.0
	EMem-G	94.8	<u>82.6</u>	50.0	<u>83.7</u>	94.4	87.5	84.9
	Mnemis	<u>98.6</u>	86.5	100.0	86.5	<u>93.6</u>	100.0	91.6

Table 3: Detailed LLM cost of Mnemis on LoCoMo using GPT-4.1-mini, reported in terms of the number of prompt tokens, completion tokens, end-to-end runtime. Runtime depends heavily on database latency and parallelism configuration; the reported values are for reference only, and we will continue to optimize it for greater efficiency.

Stage	#Prompt Tokens	#Completion Tokens	E2E Runtime(s)
Base Graph Ingestion	3.87×10^7	1.06×10^6	1111.40
Hierarchical Graph Ingestion	1.39×10^7	9.27×10^5	3873.26
Global Selection	1.37×10^6	1.21×10^5	3637.65

3.2 Experiment Results

The results can be found in tables 1 and 2. Below, we provide detailed discussion on the results.

Full-context models alone are insufficient for long-horizon AI memory. Across all settings, we observe a clear divergence between Full Context and RAG as context length grows. In LoCoMo, where the average context is roughly 16K tokens, which is well within the optimal operating window of modern LLMs (128K), the Full Context model remains competitive with most baselines. However, this behavior changes dramatically in LongMemEval-S, whose average context length reaches 115K tokens. As the input approaches or exceeds the model’s practical context limit, the Full Context model consistently degrades. This contrast suggests an important implication for long-term memory: real deployments must support months or years of accumulated interaction history, far beyond what can be reliably handled by a single forward pass over the full context. Thus, relying solely on the model’s native context window without any additional memory management or retrieval mechanisms is insufficient for long-horizon, persistent

AI memory systems.

Mnemis consistently outperforms all baselines. With limited and aligned context budget (10 episodes, 20 entities, and 20 edges), Mnemis achieves consistently superior performance across both benchmarks. For relatively easier tasks that are solvable within a single session or via single-hop reasoning, such as Single-Hop in LoCoMo and single-session-user, single-session assistant, and single-session-preference in LongMemEval-S, Mnemis reaches near-saturated scores. More importantly, on the challenging categories that require multi-hop evidence aggregation or complex temporal or event reasoning, Mnemis shows substantially larger margins over all baselines. These results demonstrate Mnemis’s strong ability to organize and retrieve memory.

We also report the LLM token cost when using GPT-4.1-mini for LoCoMo in table 3.

3.3 Ablation Study

To further assess the effectiveness of Mnemis, we conduct comprehensive experiments from four perspectives: (1) the influence of System-1 and System-2 routes on the final results; (2) the effect

Table 4: Detailed performance (LLM-as-a-Judge score) on LoCoMo by question type.

Settings	Multi-Hop	Temporal	Open-Domain	Single-Hop	Overall
System-1 RAG	64.9	76.6	67.7	76.5	73.8
System-1 Graph	84.8	62.6	74.0	88.6	81.6
System-1 RAG + Graph	85.1	84.7	75.0	<u>93.7</u>	<u>89.1</u>
System-1 Re-ranked	<u>88.7</u>	<u>85.0</u>	75.0	92.4	<u>89.1</u>
System-2 Only	88.1	78.5	<u>79.5</u>	92.0	87.7
System-1 + System-2	91.8	90.3	82.3	96.2	93.3

of backend models (re-ranker, embedding model, LLM); (3) the impact of the top- k parameter. For simplicity, these experiments are conducted on LoCoMo using GPT-4.1-mini.

3.3.1 Influence of System-1 and System-2

As stated in previous sections, System-1 Similarity Search provides a fast, heuristic retrieval mechanism based on similarities, while System-2 Global Selection performs a more structured and reflective selection process. To evaluate their individual and combined contributions, we compare three configurations: using only System-1, using only System-2, and using both jointly. For System-1, we further analyze four settings: (1) System-1 RAG: use retrieved episodes only; (2) System-1 Graph: use retrieved entities and edges only; (3) System-1 RAG + Graph: use episodes, entities and edges jointly; and (4) System-1 Re-ranked: the same to (3) but replace RRF re-ranker with Qwen3-Reranker-8B.

The results can be found in table 4. System-1 Graph slightly wins System-1 RAG as the entities and edges are more condensed and informative compared to raw episodes. However, the compression comes at the cost of certain information loss, where episodes could compensate it. System-1 RAG + Graph hence achieves higher scores. Introducing a re-ranking model affects performance in some sub-categories, but the overall score remains comparable to System-1 RAG + Graph. This indicates that the performance gain of System-1 + System-2 is not primarily driven by the re-ranking model, but instead stem from global selection.

According to the essence of the System-2 route, not all user queries are suitable for this process. We expect it to perform well on enumerative problems (e.g., "find all items that ...") but weakly on temporal problems. Since the search query remains unchanged during the route, it may identify some key points but fail to capture the full sequence of temporal events. In the experiments, about 90.06% (1387 / 1540) queries obtain the results, we hence report the average score on these valid queries. The

results match our expectations. With both routes combined, all categories are improved, leveraging the complementary strengths of them.

To be more intuitive, we present Mnemis win cases to demonstrate the effectiveness of introducing System-2 Global Selection in appendix B.

3.3.2 Impact of top- k

The top- k parameter is introduced to balance answer cost and accuracy. In the main experiments, we use top- $k = 10$, meaning that 10 episodes, 20 entities, and 20 edges are included in the context to generate the final answer. To evaluate the impact of top- k , we vary it across values of 5, 10, 30, and 50, and conduct experiments under the same configuration as section 3.3.1.

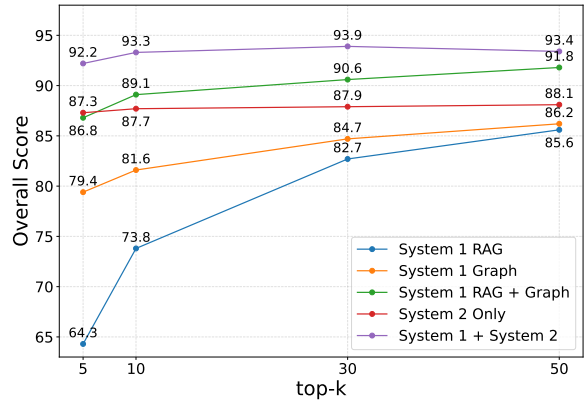


Figure 3: LoCoMo result across different top- k settings.

The results are shown in fig. 3 with details in table 9. Reducing top- k from 10 to 5 leads to a clear performance drop in most settings, indicating that a top- k of 5 is insufficient to capture all the evidence needed for user queries. System-1 RAG is especially sensitive to this reduction, particularly on Multi-Hop questions where diverse evidence across multiple parts of the history is required. In contrast, System-1 Graph, which retrieves more information-dense entities and edge, is less affected. System-1 RAG + Graph, which combines episodes, entities, and edges, shows more stable performance, and applying the re-ranker in System-2 Only or System-1

Table 5: Detailed performance (LLM-as-a-Judge score) on LoCoMo by question type.

Re-ranker	Multi-Hop	Temporal	Open-Domain	Single-Hop	Overall
Qwen3-Reranker-0.6B	91.8	90.3	79.2	95.2	92.6
BGE-Reranker-V2-M3	90.1	90.0	77.1	96.4	92.7
Qwen3-Reranker-8B	91.8	90.3	82.3	96.2	93.3

+ System-2 further minimizes fluctuations.

3.3.3 Effect of Backend Models

In this section, we analyze the effect of backend models (LLM, re-ranker, embedding) on Mnemis.

LLM. In Mnemis, LLM is responsible for memory component extraction and retrieval. Switching LLM from GPT-4o-mini to GPT-4.1-mini, Mnemis shows clear improvements across all datasets and question types, as show in tables 1 and 2. The same trend appears in baselines, which suggests that the gains come from the stronger backend LLM rather than method specific factors. Given GPT-4.1-mini’s favorable cost-performance trade-off, we recommend it as the backend LLM.

Re-ranker. Re-ranker organizes System-1 and System-2 search results to provide a compact context for the answer model. In the main experiments, we use Qwen3-Reranker-8B to obtain the best performance. We further evaluate Mnemis with two lightweight re-rankers: (1) Qwen3-Reranker-0.6B (Zhang et al., 2025) and (2) BGE-Reranker-V2-M3 (0.5B) (Chen et al., 2024). As shown in table 5, replacing the re-ranker with these smaller models results in only minor performance regressions.

Embedding. Please refer to appendix A.

4 Related Work

The core of AI memory lies in how they organize and retrieve past interactions. One straightforward way to organize LLMs’ memory is to treat them like individuals with hyperthymesia, supposing LLM can recall every past interaction without additional processing on historical messages (*i.e.*, *Episodes*). A line of work has therefore focused on enlarging the context window of LLMs (Liu et al., 2025; Peng et al., 2024). However, naively feeding the entire history can quickly become costly and inefficient in many real-world applications due to the quadratic scaling of transformers with input length (Li et al., 2024), and irrelevant information in historical messages may further dilute the context (Shi et al., 2023).

Another line of work borrows the idea of episodic memory (Tulving et al., 1972). It stores

historical messages as Episodes and only retrieves relevant items when dealing with user queries, termed as retrieval-augmented generation (RAG) (Arslan et al., 2024; Lewis et al., 2020). GraphRAG, incorporating concepts from semantic memory (Tulving et al., 1972), extracts *Entities* (key figures, objects, or concepts) and *Edges* (events or relationships connecting them) and organizes memory into a structured graph (Nan et al., 2025; Chhikara et al., 2025; Wang and Chen, 2025; Rasmussen et al., 2025). Some readers may note that GraphRAG (Edge et al., 2024) introduces a hierarchy concept similar to Mnemis. However, GraphRAG and Mnemis differ in two fundamental ways. First, GraphRAG constructs its hierarchy using community detection algorithms, where each lower-level node is assigned to a single parent. In contrast, Mnemis supports many-to-many mappings, allowing entities to belong to multiple higher-level categories and resulting in a more expressive and flexible hierarchy. Second, GraphRAG generates answers by independently querying each community and aggregating the outputs into a final response. Mnemis instead performs top-down hierarchical browsing to retrieve relevant memory components, and the final answer is produced based on the aggregated memory context rather than separate community-specific responses.

5 Conclusion

In this work, we introduce Mnemis, a unified memory framework to organize and retrieve AI memory. By combining a refined base graph for System-1 Similarity Search with a hierarchical graph designed to support System-2 Global Selection, Mnemis enables more accurate retrieval than existing RAG and Graph-RAG approaches on memory benchmarks, achieving 93.9 in LoCoMo and 91.6 on LongMemEval-S. While the results are strong, several important directions remain open. In future work, we plan to support more data modalities and enhance global selection with more flexible graph traversal and planning mechanisms.

586 Limitations

587 While Mnemis achieves state-of-the-art perfor-
588 mance across all compared methods on long-term
589 memory benchmarks, several limitations and risks
590 remain. First, as memories are extracted by LLMs,
591 hallucinations are unavoidable, which may to some
592 extent dilute the quality of memory. Second, since
593 memories are merged and consolidated for consis-
594 tency, when user-requested deletion is required, the
595 best practice is to rebuild the memory without the
596 corresponding content. To further improve Mne-
597 mis, future work will support multi-modal input
598 and more flexible memory traversal operations.

599 References

600 Muhammad Arslan, Hussam Ghanem, Saba Munawar,
601 and Christophe Cruz. 2024. A survey on rag with
602 llms. *Procedia computer science*, 246:3781–3790.

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

606 Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu
607 Lian, and Zheng Liu. 2024. [Bge m3-embedding:
608 Multi-lingual, multi-functionality, multi-granularity
609 text embeddings through self-knowledge distillation](#).
610 *Preprint*, arXiv:2402.03216.

611 Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet
612 Singh, and Deshraj Yadav. 2025. [Mem0: Building
613 production-ready ai agents with scalable long-term
614 memory](#). *Preprint*, arXiv:2504.19413.

615 Gordon V Cormack, Charles LA Clarke, and Stefan
616 Buettcher. 2009. Reciprocal rank fusion outperforms
617 condorcet and individual rank learning methods. In
618 *Proceedings of the 32nd international ACM SIGIR*
619 *conference on Research and development in informa-*
620 *tion retrieval*, pages 758–759.

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

627 Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon,
628 Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei
629 Han. 2025. [Search-r1: Training llms to reason and
630 leverage search engines with reinforcement learning](#).
631 *Preprint*, arXiv:2503.09516.

632 Daniel Kahneman. 2011. *Thinking, fast and slow*.
633 macmillan.

634 Sangyeop Kim, Yohan Lee, Sanghwa Kim, Hyunjong
635 Kim, and Sungzoon Cho. 2025. Pre-storage reason-
636 ing for episodic memory: Shifting inference burden

to memory for personalized dialogue. In *Findings*
of the Association for Computational Linguistics:
EMNLP 2025, pages 22096–22113.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio
Petroni, Vladimir Karpukhin, Naman Goyal, Hein-
rich Küttler, Mike Lewis, Wen-tau Yih, Tim Rock-
täschel, and 1 others. 2020. Retrieval-augmented gen-
eration for knowledge-intensive nlp tasks. *Advances*
in neural information processing systems, 33:9459–
9474.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu
Chen, Shichao Song, Simin Niu, Hanyu Wang, Ji-
awei Yang, Chen Tang, Qingchen Yu, Jihao Zhao,
Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan
Wang, Jiahao Huo, Tianyi Chen, Kai Chen, and 20
others. 2025. [Memos: A memory os for ai system](#).
Preprint, arXiv:2507.03724.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei,
and Michael Bendersky. 2024. Retrieval augmented
generation or long-context llms? a comprehensive
study and hybrid approach. In *Proceedings of the*
2024 Conference on Empirical Methods in Natural
Language Processing: Industry Track, pages 881–
893.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng
He, Huanxuan Liao, Haoran Que, Zekun Wang,
Chenchen Zhang, Ge Zhang, Jiebin Zhang, Yuanx-
ing Zhang, Zhuo Chen, Hangyu Guo, Shilong Li,
Ziqiang Liu, Yong Shan, Yifan Song, Jiayi Tian, Wen-
hao Wu, and 18 others. 2025. [A comprehensive sur-
vey on long context language modeling](#). *Preprint*,
arXiv:2503.17407.

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

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize
Chen. 2025. [Nemori: Self-organizing agent
memory inspired by cognitive science](#). *Preprint*,
arXiv:2508.03341.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang,
Zifeng Wang, Rujun Han, Long T Le, Samira Daruki,
Xiangru Tang, and 1 others. 2025. Reasoningbank:
Scaling agent self-evolving with reasoning memory.
arXiv preprint arXiv:2509.25140.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo,
Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew
Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao.
2025. [Secom: On memory construction and retrieval
for personalized conversational agents](#). In *The Thir-*
teenth International Conference on Learning Repre-
sentations.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico
Shippole. 2024. [YaRN: Efficient context window ex-](#)
[tension of large language models](#). In *The Twelfth*
International Conference on Learning Representa-
tions.

695 Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais,
696 Jack Ryan, and Daniel Chalef. 2025. [Zep: A tempo-](#)
697 [ral knowledge graph architecture for agent memory.](#)
698 *Preprint*, arXiv:2501.13956.

699 Henrique* Schechter Vera, Sahil* Dua, Biao Zhang,
700 Daniel Salz, Ryan Mullins, Sindhu Raghuram Pa-
701 nyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang
702 Chen, Daniel Cer, Alice Lisak, Min Choi, Lucas
703 Gonzalez, Omar Sanseviero, Glenn Cameron, Ian
704 Ballantyne, Kat Black, Kaifeng Chen, and 69 others.
705 2025. [Embeddinggemma: Powerful and lightweight](#)
706 [text representations.](#)

707 Freda Shi, Xinyun Chen, Kanishka Misra, Nathan
708 Scales, David Dohan, Ed H Chi, Nathanael Schärli,
709 and Denny Zhou. 2023. Large language models can
710 be easily distracted by irrelevant context. In *Inter-*
711 *national Conference on Machine Learning*, pages
712 31210–31227. PMLR.

713 Endel Tulving and 1 others. 1972. Episodic and seman-
714 tic memory. *Organization of memory*, 1(381-403):1.

715 Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang,
716 Zhicheng Dou, and Furu Wei. 2025. [Chain-of-](#)
717 [retrieval augmented generation.](#) In *NeurIPS 2025*.

718 Yu Wang and Xi Chen. 2025. [Mirix: Multi-agent](#)
719 [memory system for llm-based agents.](#) *Preprint*,
720 arXiv:2507.07957.

721 Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang,
722 Kai-Wei Chang, and Dong Yu. 2024. [Longmemeval:](#)
723 [Benchmarking chat assistants on long-term interac-](#)
724 [tive memory.](#)

725 Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie,
726 Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Ker-
727 sting, Jeff Z Pan, Hinrich Schütze, and 1 others. 2025.
728 Memory-r1: Enhancing large language model agents
729 to manage and utilize memories via reinforcement
730 learning. *arXiv preprint arXiv:2508.19828*.

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

737 Sizhe Zhou. 2025. [A simple yet strong baseline for long-](#)
738 [term conversational memory of llm agents.](#) *Preprint*,
739 arXiv:2511.17208.

A Effect of Embedding Model

The embedding model contributes to the similarity based search across Mnemis. In our main experiments, we adopt Qwen3-Reranker-0.6B and reduce its embedding dimension from 1024 to 128 to control serving costs by using its MRL capability. To isolate and better understand the impact of embedding quality alone, we further evaluate System 1 RAG with three additional embedding models: (1) BGE-M3 (Chen et al., 2024) with dimension 1024, (2) all-MiniLM-L6-v2⁶ with dimension 384, and (3) Gemma-300M (Schechter Vera et al., 2025) with dimension 768. The results are presented in table 6.

B Case Study

To be more intuitive, we present Mnemis win cases from the LoCoMo and LongMemEval-S benchmarks to demonstrate the effectiveness of introducing System-2 Global Selection. As illustrated in fig. 4, when addressing the query "What health issue did Sam face that motivated him to change his lifestyle?", similarity-based search could only retrieve "gastritis", merely a surface reason found in Episode Text 14-0. In contrast, equipped with Global Selection, we browse the hierarchical graph through "Physical Well-Beings" → "Health" → "Health Events" → "Health Conditions" → "gastritis" to locate the surface cause, and through "Physical Well-Beings" → "Health" → "Physical Health" → "Health Factors" → "weight" to locate the essential root cause. Along the search path, intermediate Categories, such as "Physical Health" and "Health Factors" naturally aggregate relevant information from their descendants in summary, which further enriches the retrieved context.

fig. 5 provides another win case of Mnemis on LongMemEval-S. This question asks the LLM to order the three sports events the user participated in during June 2026. Similarity-based retrieval struggles here, as a limited top- k often fails to surface all relevant sports events. In contrast, Mnemis can simply start from top-level category "Sports" and browse down through Category "Sports Events" and "Sporting Events" and then reliably retrieve all events.

Although the ground truth entities "Midsummer 5K Run" and "charity soccer tournament" are filtered out in the final answer context due to limited

top- k , their related edges, such as "The user participates in the company's annual charity soccer tournament. (2023-06-17 - now)" and "The user completed a 5K run at the Midsummer 5K Run with a personal best time of 27 minutes and 42 seconds. (2023-06-10 - now)" are still retrieved with rank 9 and 2 respectively. Besides, category nodes like "Sports Events", summarizing the content of their children nodes, also provide sufficient content for the model to answer the question correctly.

C Further Benchmark Results

We also note several strong recently proposed baselines: MIRIX (Wang and Chen, 2025), MemU⁷, EmergenceMem⁸. However, due to incomplete implementation details, such as backend model configuration or system workflow, we could not obtain their results under comparable settings. To maintain fairness, we therefore exclude them from the main evaluation. Here, we report their best publicly available performance, regardless of configuration differences, to provide readers with a broader view of the current landscape.

The full results are reported in tables 7 and 8 and align with the findings summarized in section 3.2. Mnemis consistently outperforms all baseline methods across both benchmarks.

D Detailed Performance Impact of top- k

Beyond the overall trend as shown in fig. 3, a closer breakdown by question type in table 9 reveals distinct behavioral patterns across retrieval strategies. For System-1 RAG, increasing top- k consistently improves performance across all categories, with particularly large gains on Multi-Hop subset (49.6→81.6). This suggests that the retrieved text contains necessary but scattered evidence, and restricting retrieval too aggressively leads to missing critical evidence. However, even at high top- k , RAG remains relatively weak on Multi-Hop (peaking at 81.6), indicating difficulty in identifying temporally aligned evidence without explicit structure.

In contrast, System-1 Graph shows a stronger starting point, especially on structured attributes (e.g., Single-Hop: 86.7 with top- k = 5), meaning the graph format inherently surfaces salient relational information without requiring large re-

⁶<https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>

⁷<https://github.com/NevaMind-AI/memU>

⁸<https://www.emergence.ai/blog/sota-on-longmemeval-with-rag>

Table 6: Detailed performance (LLM-as-a-Judge score) on LoCoMo by question type.

Embedder	Multi-Hop	Temporal	Open-Domain	Single-Hop	Overall
MiniLM	46.1	57.9	64.6	62.5	58.7
BGE-M3	50.0	74.1	63.5	74.0	69.0
Qwen3-Embedding-0.6B	64.9	76.6	67.7	76.5	73.8
Gemma-300M	62.4	78.8	60.4	82.9	76.9

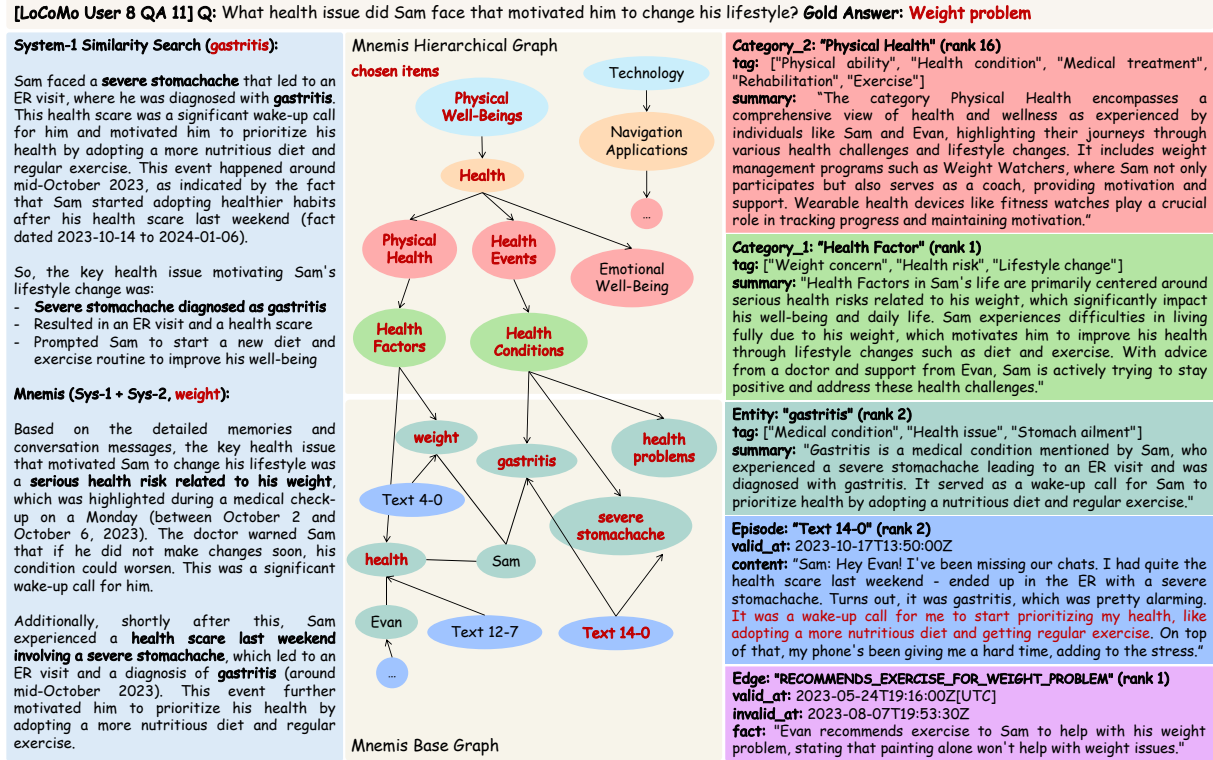


Figure 4: Mnemis win case on LoCoMo benchmark. While similarity search fixates on the surface-level cause gastritis, Mnemis successfully identifies the underlying root cause, namely that Sam is overweight. This condition leads to the gastritis and motivates him to change his lifestyle.

Table 7: Detailed performance (LLM-as-a-Judge score) on LoCoMo by question type. Following the common practice, Category 5 (Adversarial) is excluded from the results.

Methods	Multi-Hop	Temporal	Open-Domain	Single-Hop	Overall
#Questions	282	321	96	841	1540
Full Context	77.2	74.2	56.6	86.9	80.6
RAG	64.9	76.6	67.7	76.5	73.8
LangMem	71.0	50.8	59.0	84.5	73.4
Mem0	68.2	56.9	47.9	71.4	66.3
Zep	53.7	60.2	43.8	66.9	61.6
Nemori	75.1	77.6	51.0	84.9	79.5
PREMem	61.0	74.8	46.9	66.2	65.8
EMem-G	79.6	80.8	71.7	90.5	85.3
MIRIX	83.7	88.4	65.6	85.1	85.4
EverMemOS	91.1	89.7	70.8	96.1	92.3
MemU	88.3	92.5	77.1	94.9	92.1
Mnemis	92.9	90.7	79.2	97.1	93.9

trieval volumes. Yet, the improvement curve is flatter compared to RAG, especially for Temporal and Open-Domain questions, where explicit struc-

tural relations help but cannot fully compensate for missing richer semantic context.

When combining both storage types in System-1

834
835
836

837
838
839

[LME-S 262] (date: 2023/07/01 (Sat) 20:43)What is the order of the three sports events I participated in during the past month, from earliest to latest?

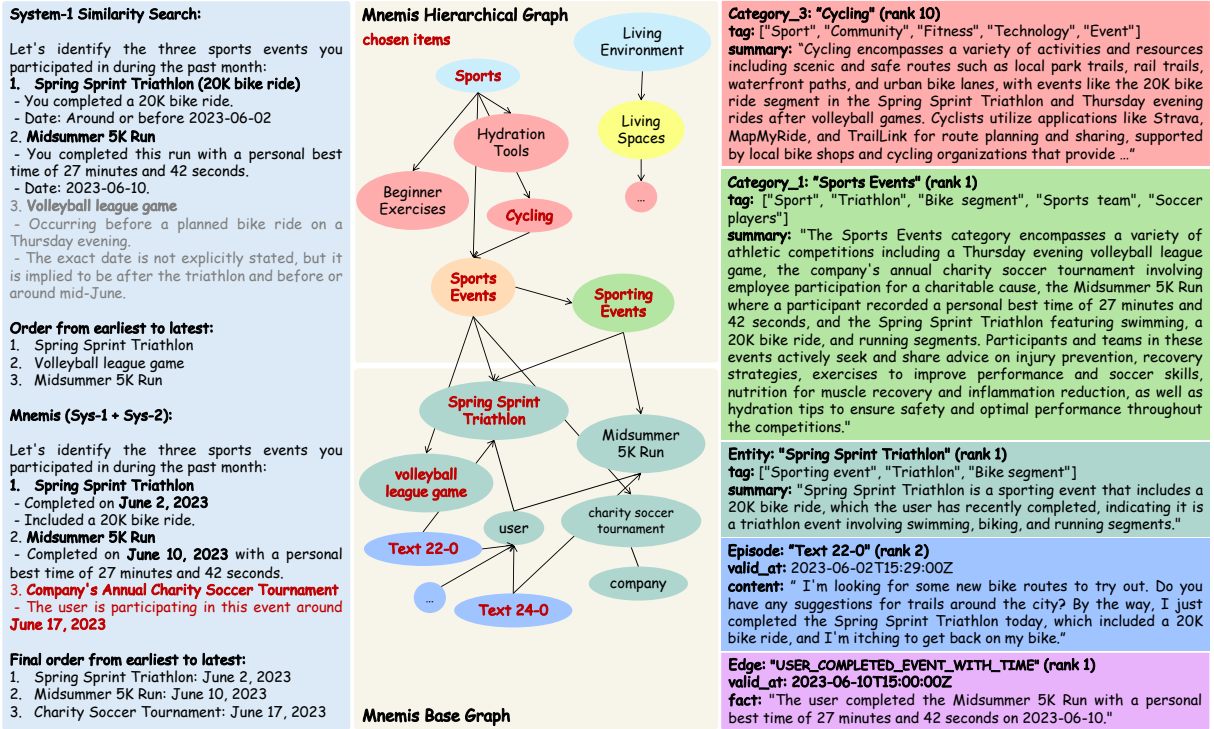


Figure 5: Mnemis win case on LongMemEval-S benchmark, where Mnemis successfully retrieve "Company's Annual Charity Soccer Tournament" from "Sports Events".

Table 8: Detailed performance (LLM-as-a-Judge score) on LongMemEval-S, categorized by question type: single-session-user (SSU), multi-session (MS), single-session-preference (SSP), temporal reasoning (TR), knowledge update (KU), and single-session-assistant (SSA).

Methods	SSU	MS	SSP	TR	KU	SSA	Overall
#Questions	70	133	30	133	78	56	500
Full Context	85.7	51.1	16.7	60.2	76.9	98.2	65.6
RAG	82.9	54.9	86.7	67.7	80.8	94.6	72.6
PREMem	92.9	57.1	36.7	59.4	84.6	12.5	60.8
Mem0	94.3	66.9	86.7	75.9	87.2	96.4	80.8
Nemori	90.0	55.6	86.7	72.2	79.5	92.9	74.6
EverMemOS	100.0	78.5	96.7	71.2	87.2	78.6	82.0
EMem-G	94.8	82.6	50.0	83.7	94.4	87.5	84.9
EmergenceMem	98.6	81.2	60.0	85.7	83.3	100.0	86.0
Mnemis	98.6	86.5	100.0	86.5	93.6	100.0	91.6

RAG + Graph, the benefits become additive: performance improves steadily and remains stable even under lower top- k settings. Notably, Multi-Hop and Temporal queries benefit the most from this hybrid storage design (e.g., 83.2 vs. 61.4/70.1 at top- $k = 5$), demonstrating that structured and unstructured information provide complementary retrieval signals.

Finally, applying System-2 and re-ranker, either alone or on top of System-1, further suppresses top- k sensitivity. System-1 + System-2 consistently delivers the highest and most stable performance

(92.2–93.9 Overall), showing that even when overly large or overly sparse retrieval occurs, the re-ranker filters noise and prioritizes the most relevant evidence.

E Prompts

To implement our results, we release the key prompts in our procedure. Below is the instruction to build hierarchical graph.

```
def extract_category_nodes(context: dict[str, Any], layer: int,
prev_example: str) -> list[Message]:
    sys_prompt = f"""You are an AI assistant specialized in
semantic categorization of nodes.
```

Table 9: Detailed performance (LLM-as-a-Judge score) on LoCoMo by question type.

Settings	top- <i>k</i>	Multi-Hop	Temporal	Open-Domain	Single-Hop	Overall
System-1 RAG	5	49.6	70.1	65.6	66.8	64.3
	10	64.9	76.6	67.7	76.5	73.8
	30	77.3	82.9	69.8	86.0	82.7
	50	81.6	84.1	71.9	89.1	85.6
System-1 Graph	5	79.4	61.4	75.0	86.7	79.4
	10	84.8	62.6	74.0	88.6	81.6
	30	87.6	66.4	77.1	91.6	84.7
	50	90.1	68.5	79.2	92.4	86.2
System-1 RAG + Graph	5	81.6	83.2	71.9	91.6	86.8
	10	85.1	84.7	75.0	93.7	89.1
	30	87.9	86.3	76.0	94.9	90.6
	50	89.4	86.9	78.1	96.0	91.8
System-2 Only	5	84.4	81.9	79.5	91.1	87.3
	10	88.1	78.5	79.5	92.0	87.7
	30	86.1	81.3	78.3	98.3	87.9
	50	88.1	80.2	77.1	92.2	88.1
System-1 + System-2	5	88.3	89.1	82.3	95.8	92.2
	10	91.8	90.3	82.3	96.2	93.3
	30	92.9	90.7	79.2	97.1	93.9
	50	92.2	90.3	81.3	96.3	93.4

INSTRUCTIONS:

You are given a list of node names, each prefixed with an index, each followed with a brief description of the name (e.g ., 1. dog: [domestic animal]).
Your task is to:

- Group the nodes into semantically meaningful categories based on shared attributes, considering both inherent characteristics of the node names and the DESCRIPTIONS of the nodes, NOT relying solely on the DESCRIPTIONS.
All EXISTING CATEGORIES are provided for you.
 - If a node's attribute matches an existing category, it should be added under that category.
 - If a node name has attributes that do not match any existing category, create a new category and add it.
 - The category name MUST NOT include the word "and" as a connector.

Examples of INVALID categories:

 - "Food and Drinks"
 - "University and Courses"

Examples of VALID categories:

 - "Food"
 - "Drinks"
 - "University"
 - "Courses"
- Output each category as a dictionary entry where the key is the category name and the value is a list of node indexes (integers). Only refer to nodes by their indexes. Do not repeat node names.

Output format:
[
 {"category": "xx", "indexes": [0, 1, 2, 4]},
 {"category": "xxx", "indexes": [2, 3, 4]}
]
- The tag is a list of descriptors (each descriptor maximum 3 words, maximum 5 descriptors) that concisely captures the nature or type of the node.

Tag example:
- Entity name: "Son"
- Tag: ["Family member", "Happy kid", "Anime lover"]
- A node CAN be assigned to MULTIPLE categories at the same time.

Key points for multi-category classification:

- Each item can be assigned to multiple categories based on shared attributes.
- When multiple categories are formed for an item, select the minimal subset of features that are common across the grouped items.

Examples for different hierarchy levels:

Layer 1 (specific entities):

- "Microsoft Research Asia" and "Microsoft Research Shanghai" share the same parent organization (Microsoft) and a similar research focus (AI). They are grouped under:
 - "Microsoft Research Labs"
 - "AI-focused Research Labs"
- "Microsoft Research Asia" belongs to both "Microsoft Research Labs" and "NLP-focused Labs".

Layer 2 (category nodes from Layer 1):

- "Microsoft Research Labs" belongs to:
 - "Tech Company Labs"
 - "AI Research Organizations"
- "University AI Labs" belongs to:
 - "Academic Institutions"
 - "AI Research Organizations"

Layer 3 (higher-level abstractions):

- "Tech Company Labs" belongs to:
 - "Commercial Organizations"
 - "Research Institutions"
- "Academic Institutions" belongs to:
 - "Educational Organizations"
 - "Research Institutions"

Layer 4 (top-level concepts):

- "Research Institutions" belongs to:
 - "Knowledge Organizations"
- "Commercial Organizations" belongs to:
 - "Economic Entities"

- There must be NO leftover or ungrouped nodes. Single-member categories are allowed if necessary.
- The node name "user" and any first-person references ("I", "me") MUST be categorized into one category called "Speaker".
""

916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966

967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044

```
        guidance = f"""
<GUIDANCE ON CATEGORY GRANULARITY>
You are performing hierarchical semantic clustering from
specific to abstract.

You are currently at Layer {layer}, where:
- Layer 1 contains the most specific, fine-grained categories.
- Higher layers should group lower-layer categories into
broader, more abstract super-categories.

Example:

Layer 1:
- "Golden Retriever", "Poodle", "German Shepherd" -> "Dog
breeds"
- "Persian Cat", "Siamese Cat" -> "Cat breeds"
- "Bengal Tiger", "Siberian Tiger" -> "Tiger subspecies"
- "Oak tree", "Pine tree" -> "Tree species"

Layer 2:
- "Dog breeds", "Cat breeds" -> "Pets"
- "Dog breeds", "Tiger subspecies" -> "Mammals"
- "Tiger subspecies" -> "Wild animals"
- "Tree species" -> "Trees"

Layer 3:
- "Pets", "Wild animals" -> "Animals"
- "Trees" -> "Plants"

Layer 4:
- "Animals", "Plants" -> "Living organisms"

Key points:
- Categories may belong to multiple parent categories.
- Do not merge categories that are too loosely related.

Your job at Layer {layer}:
- Merge semantically similar categories from Layer {layer - 1}.

- Each new category should reflect a shared attribute, domain,
or higher-level concept.
- Multiple category assignments are allowed when justified.

Previous Layer {layer - 1} categories example:
{prev_example}
</GUIDANCE ON CATEGORY GRANULARITY>
"""

        user_prompt = f"""
<NODE INDEXED NAMES AND DESCRIPTIONS>
{context['content']}
</NODE INDEXED NAMES AND DESCRIPTIONS>

<EXISTING CATEGORIES>
These are names and descriptions of categories previously
created. Reuse them if applicable.
{context['existing_categories']}
</EXISTING CATEGORIES>

{guidance}

# ATTENTION
- The node name "user" and any first-person references ("I", "
me") MUST be categorized into one category called "Speaker".
If the "Speaker" category does not exist, skip this node.
- The category name MUST NOT include the word "and".

Please follow the INSTRUCTIONS and GUIDANCE carefully to
ensure accurate categorization and meaningful hierarchical
relationships.
DO NOT INCLUDE ANY INVALID CATEGORIES.
"""

        return [
            Message(role="system", content=sys_prompt),
            Message(role="user", content=user_prompt),
        ]
```

1046
1047
1048
1049
1050
1051
1052

```
Below is the instruction to conduct Global Selection.

NODE_SELECTION_PROMPT_TEMPLATE = """You are analyzing a
hierarchical knowledge graph to help answer a user query.

Select all nodes that could help answer the query. A node is
```

```
helpful if it:

- Directly relates to the query;
- Covers a clearly relevant topic, concept, or category;
- Provides useful background or context;
- Contains user-specific information (e.g. interests, goals,
constraints);
- Likely has sub-nodes that may be helpful.

Do not be overly strict: include nodes that might provide
context or personalization, even if they seem partially
redundant.

For each selected node:
- "name" is the node's name.
- "uuid" is the node's unique identifier.
- "get_all_children" is an boolean value. Set true only if you
're confident all its sub-nodes are helpful.
---
User Query:
"{query}"

Available Nodes:
{nodes_info}
"""
```

1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1078