Does Memory Need Graphs? A Unified Framework and Empirical Analysis for Long-Term Dialog Memory

Sen Hu[1,†,*], **Yuxiang Wei**[2,†], **Jiaxin Ran**[1,†],
Zhiyuan Yao[3],**Xueran Han**[4], **Huacan Wang**[5], **Ronghao Chen**[1], **Lei Zou**[1]
[1]Peking University [2]Georgia Tech [3]ZJU [4]MBZUAI [5]UCAS
† Equal contribution * Corresponding author: husen@pku.edu.cn

Abstract

Graph structures are increasingly adopted in dialog memory systems, motivated by their success in retrieval-augmented generation and the associative nature of human memory. However, empirical findings on their effectiveness remain inconsistent, making it unclear which design choices truly matter. In this work, we present an experimental and system-oriented analysis of long-term dialog memory architectures. We formalize a unified framework that decomposes dialog memory systems into core components and supports both graph-based and non-graph approaches. Under this framework, we conduct controlled, stage-wise experiments on LongMemEval and HaluMem, comparing common design choices in memory representation, organization and maintenance, as well as indexing and retrieval. Our results show that underlying implementation details—often insufficiently specified in prior work—have a substantial impact on performance, and we identify stable, reliable strong baselines to support fair comparison and practical deployment. Code are available at https://github.com/AvatarMemory/UnifiedMem.

1 Introduction

With the widespread adoption of graph structures in retrieval-augmented generation (RAG) systems (Peng et al., 2024; Zhang et al., 2025a), and the rapid development of research on long-term dialog memory, incorporating graph structures into dialog memory systems has gradually become a common design choice (Hu et al., 2025). From an intuitive perspective, human memory is not stored as isolated units but organized through associative relationships (Wu et al., 2025a), which makes explicit relational modeling a natural conceptual fit for dialog memory.

Despite this intuition, existing literature reveals that empirical conclusions on dialog memory methods remain far from consistent. Some studies (Chhikara et al., 2025; Zhang et al., 2025c) report performance gains from introducing relatively complex graph construction and graph-based retrieval mechanisms, while other works (Wu et al., 2025b; Fang et al., 2025) show that non-graph paradigms or lightweight structures achieve comparable or even better results on the same or similar benchmarks. In the absence of a unified analytical perspective, such discrepancies are difficult to attribute to specific design choices, which complicates both the interpretation of prior results and the practical decision-making process for system builders.

We observe that comparing and attributing existing results remains difficult for several reasons. Different works often adopt different datasets, backbone models, or evaluation settings, and underlying implementation details are sometimes insufficiently specified, limiting reproducibility. More fundamentally, dialog memory research lacks a unified system-level framework that covers both graph-based and non-graph approaches.

As a result, method-centric studies often describe end-to-end systems around a small set of highlighted innovations, while other system components—such as memory representation, indexing, retrieval and ranking—are implemented with varying assumptions and levels of specificity across methods. When these foundational settings are not aligned, performance differences become difficult to attribute to specific design choices, and reported results across studies can be difficult to interpret or reconcile [1].

Based on these observations, this paper is positioned as an **experimental and system-oriented analysis** rather than a proposal of a new method. Our contributions are summarized as follows:

- Inspired by LongMemEval, we formalize a unified framework for dialog memory that encompasses both graph-based and non-graph

[1]More details can be found in Appendix D

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 26758–26782

methods, providing a foundation for aligning and comparing different approaches.

- Using the LongMemEval and HaluMem benchmarks, we conduct controlled, stage-wise experimental comparisons of several commonly adopted design choices in dialog memory systems, covering memory unit design, memory organization and maintenance mechanisms, as well as graph-based and non-graph indexing and retrieval strategies.
- Through fine-grained empirical analysis, we identify and validate a set of stable and reliable strong baselines that hold across both graph-based and non-graph approaches. These baselines can serve as a common starting point for future dialog memory research and system development, lowering the barrier to fair comparison and practical deployment.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) retrieves relevant information from a pre-constructed external knowledge base to enrich the prompt, thereby enabling large language models to produce more grounded and accurate responses (Lewis et al., 2020). However, traditional RAG approaches struggle with understanding complex queries and with the sparsity of domain-specific knowledge in pre-built corpora. While techniques like query rewriting, retrieval-time enhancement (Mao et al., 2021), and synchronized retrieval (Jiang et al., 2024) can help, they remain fundamentally constrained by chunk-based textual knowledge organization.

To address these limitations, recent work has introduced graph-structured knowledge into the retrieval process. GraphRAG (Edge et al., 2024) leverages graph topology to capture rich entity relationships, enabling relationship-driven retrieval and multi-hop reasoning. Building on this line of work, HippoRAG2 (Gutiérrez et al., 2025) adopts a hybrid hierarchical graph and performs structure-aware retrieval via graph-based propagation. LightRAG (Guo et al., 2024) proposes a dual-level graph retrieval mechanism that enhances coverage and reasoning by jointly leveraging fine-grained local knowledge and higher-level global structure. Collectively, these methods point toward a more structured direction for future memory systems in intelligent agents, improving both knowledge organization and reasoning capability.

2.2 Agent Memory Augmentation

Memory plays a crucial role in agent systems, serving as a key component that enables agents to sustain long-term attention, dynamically acquire new knowledge from historical data, and act autonomously (Qing et al., 2026; Diao et al., 2025; Tang et al., 2026). In the context of long-term dialog agents, research on memory augmentation can be broadly divided into two categories: flat-index-based and graph-index-based methods. Wu et al. experimentally demonstrated that existing LLMs and commercial chat assistants exhibit significant performance degradation in handling long-term dialog memory, leading them to propose the LME benchmark (Wu et al., 2024). RMM (Tan et al., 2025) effectively transcends the limitations of traditional fixed-length chunk boundaries through topic-driven dynamic summarization. Beyond these flat index-based approaches, Mem0-g (Chhikara et al., 2025) represents memory as a labeled graph where entities act as nodes and relations act as edges. Zep and CAM (Rasmussen et al., 2025; Li et al., 2025) introduce a label propagation algorithm to mitigate the limitations of traditional GraphRAG (Edge et al., 2024) in handling incremental updates of communities.

2.3 Difference between RAG and Memory

RAG and agent memory systems, despite their separate origins, now employ substantially overlapping techniques (Hu et al., 2025). RAG was initially conceived to connect LLMs to static knowledge sources, whereas agent memory prioritized knowledge updating and persistence. As both fields advance, their methodologies increasingly overlap. For instance, works like HippoRAG (Jimenez Gutierrez et al., 2024; Gutiérrez et al., 2025) are now recognized in both RAG and memory research as solutions to the long-term memory challenges of large language models.

Empirically, RAG tasks typically build indexes directly over raw content such as documents and do not require frequent updates, whereas dialog memory systems tend to index extracted "memories" from conversations and place greater emphasis on online index maintenance. Drawing on both research traditions, we propose a unified framework tailored for long-term dialogue memory.

	Key	Value	Query	Index (Struct; Op)	Retrieval	Answering
LongMemEval	Session + Facts	Session	Q + T	F; Add	Q → K → V	CoN
RMM	Topic summ	Session + Key	Q	F; Add, Upd	Q → K → V → Rerank	Direct
A-Mem	Session + Kw, Tag, Summ	Key	Q	G; Add, Upd	Q → K → V	Direct
Mem0-G	Entity name, Triple	Triple	Q	G; Add, Align	Q → K → 1-Hop → V	Direct
Zep	Entity summ, Triple, Comm	Key	Q	HG; Add, Align, Comm Upd	Q → K → 1-Hop → V → Rerank	Direct

Table 1: Decomposition of some representative dialog memory systems under the unified framework. A more complete version and detailed explanations can be found in Appendix D.

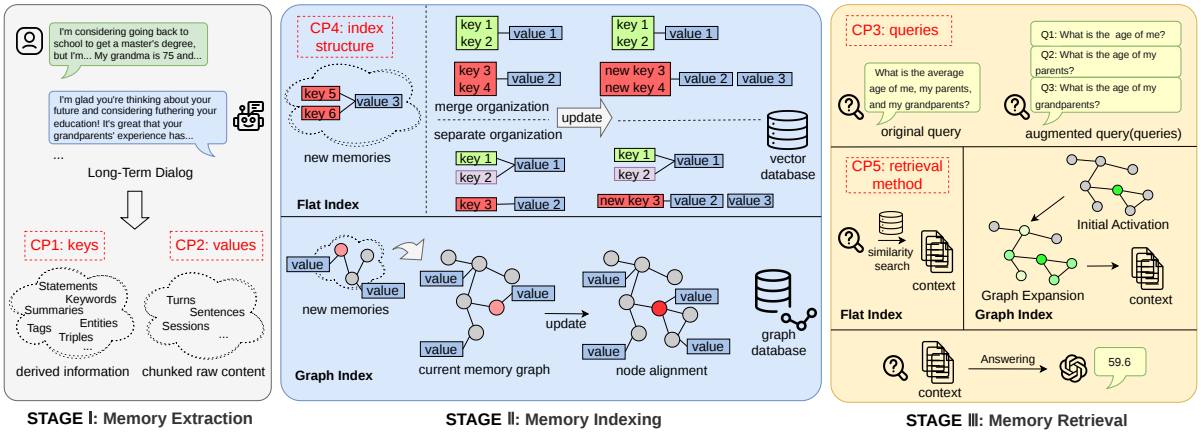


Figure 1: The unified framework for dialog memory system.

3 A Unified Memory Framework

We abstract dialogue memory and RAG-style memory systems into a six-tuple:

$$\langle K, V, Q, I, R, A \rangle,$$

where K denotes the set of *keys* (memory units), V denotes the *values* (evidence used in the answering stage), Q denotes the *queries* (retrieval requests), I denotes the *index structure* and R denotes the *retrieval method*. A denotes the answering strategy, the method that generates the final response from retrieved values.

The overall pipeline can be decomposed into four stages:

- **Memory Extraction:** extract memory from raw dialogue, i.e., generate keys and values;
- **Memory Indexing:** organize keys into a flat index or graph structure, building and maintaining I;
- **Memory Retrieval:** given a query $q \in Q$, get target keys in K using retrieval method R, map back to the corresponding values in V.
- **Question Answering:** assemble the final context and answer the user input.

In practical applications, the above four operational stages must be invoked repeatedly as the dialog progresses. The invocation timing depends on the specific strategy. For example, memory extraction and indexing may be performed at the end of each session, while memory retrieval and answering may be executed after each user input.

Table 1 shows how representative dialog memory systems can be decomposed and aligned under this unified framework, enabling comparison across otherwise heterogeneous implementations. Appendix A provides additional details for each stage.

3.1 Keys and Values

Keys usually represent important information derived from dialog history, which can be seen as memory units. Depending on the design, keys may correspond to summaries, factual statements, keywords (Xu et al., 2025), or entities and triples (Rasmussen et al., 2025), and may encode either semantic or episodic information (Du et al., 2025). As illustrated in Table 1, some existing work also considers raw content (e.g., sessions) as one form of Key (Wu et al., 2024).

Values correspond to the evidence ultimately provided to the answering model. Using the retrieved keys as values is a natural choice, and some prior work (Tan et al., 2025) additionally (or alternatively) uses the raw content as values. Raw content such as sessions preserve contextual continuity but

incur higher storage and reasoning costs, whereas derived information are usually compressed, which improve efficiency at the risk of information loss.

3.2 Query and Answering

Query Augmentation has been widely adopted in RAG systems (Guo et al., 2024; Gutiérrez et al., 2025), to improve the understanding of user intent during the retrieval stage and to enhance the recall quality of relevant documents. Common operations include query rewriting, expansion, and decomposition, among others. This paradigm has recently been extended to dialogue memory tasks in several studies (Zhang et al., 2025b). The answer generation stage can also benefit from several optimization techniques, such as extracting key information before answering (Yu et al., 2023).

Since query augmentation and answering optimization are relatively independent, they are not discussed in detail in this work, and the original input is used as the query by default.

3.3 Index Structure

The index structure specifies how keys are organized and maintained, providing the substrate on which retrieval operates.

3.3.1 Flat Index and Key Organization

In the simplest designs, each key is stored as an independent vector entry under the **Separate Organization**. The separate organization results in a large number of vectors and suffers from issues of information fragmentation and redundancy. While some keys (usually derived from same session) can be concatenated to jointly produce a single vector representation with the **Merge Organization**. More specifically, A-Mem (Xu et al., 2025) introduces the concept of Memory Notes and aggregate all keys derived from the same value into a single vector (which we refer to as the Merge-by-session). While LongMemEval further explores aggregating only keys of the same type into one vector (which we refer to as the Merge-by-type).

3.3.2 Graph Index

Graph indices connect keys via explicit edges. In graph-based designs, nodes may represent textual units or entities, while edges capture relationships such as semantic relatedness, structural association, or temporal order. Some systems further adopt hierarchical graph structures to connect multiple abstraction levels (e.g., entity–community hierarchies (Edge et al., 2024)).

3.3.3 Index Maintenance and Update

In theory, memories acquired at different times may exhibit relationships such as duplication, completion, or update and replacement. Common memory maintenance operations (Chhikara et al., 2025) include *Add*, *Update*, *Delete* and *Noop*. Recent work (Ong et al., 2025) argues that directly deleting old memories is undesirable, as they may still be useful in the future (e.g., when a user refers to a former occupation).

In practice, these operations are generally applicable to both flat index and graph-based index. Figure 1 Stage II illustrates the *Update* operation. In graph-based settings, the *Update* operation is typically referred to as node merge (also known as entity alignment) and the corresponding edge merge (relation alignment).

For hierarchical graph indices, additional maintenance operations are required, such as community reconstruction (Edge et al., 2024) or incremental updates (Rasmussen et al., 2025; Li et al., 2025).

3.4 Retrieval Procedure

3.4.1 Retrieval over Flat Index

For flat indices, retrieval typically follows a standard vector-search pipeline: embed the query q into a vector representation, then perform approximate or exact nearest-neighbor search over the key embeddings, and select top-k keys according to similarity scores. Finally map these keys back to their associated values and assemble the context.

Note that keys and values are often not in a one-to-one relationship. Therefore, to ultimately obtain the top-n values for answering a query, the value of k is typically set larger than n, and a value-level re-ranking mechanism is required.

3.4.2 Retrieval over Graph Index

Graph-based retrieval is defined as retrieving a subgraph (or some nodes) conditioned on a query q. The graph retrieval process typically combines an initial activation step with graph-based expansion.

Initial Activation Similar to flat indices, first obtain an initial set of seed nodes: embed q and perform similarity search over node embeddings, select top-k nodes. Or allow direct vector matching between Q and triples (Chhikara et al., 2025).

Graph Expansion Starting from the seed nodes, graph expansion methods aim to gather additional context from the graph. A common approach is BFS-style local expansion (Rasmussen et al., 2025), which collects nodes within a limited hop distance (typically one hop) and truncates the result when a predefined maximum number of nodes is reached. Beyond this, more advanced structure-aware (Gutiérrez et al., 2025) or semantics-aware (Zhang et al., 2025c) methods guide expansion using graph topology or embedding similarity.

4 Experiments

To understand how different design choices and underlying implementation affect the behavior of dialog memory systems, we conduct a series of controlled, stage-wise experiments under the unified framework introduced in Section 3.

The experiments are organized as follows. Sections 4.2 and 4.3 examine flat-based methods and aim to identify strong baseline configurations. Sections 4.4 and 4.5 focus on graph-based methods, analyzing graph construction strategies and retrieval mechanisms. Section 4.6 presents the end-to-end experimental results based on selected settings.

Due to space limit, we report only a subset of the results in the main paper; more detailed experimental results (including efficiency analysis) can be found in the Appendix C.

4.1 Datasets and Settings

We conduct experiments on two representative benchmarks, LongMemEval and HaluMem, which are designed to evaluate long-term dialog memory from complementary perspectives.

LongMemEval (Wu et al., 2024) primarily focuses on memory retrieval and reasoning over extremely long dialog histories and consists of two subsets (S and M).

HaluMem (Chen et al., 2025) is designed to evaluate memory extraction, updating, and consistency. HaluMem contain a large number of information updates and explicitly test whether memory systems can correctly extract new information, update outdated memories, and avoid hallucinated responses. It consists of Medium and Long subsets, due to the high computational cost, we conduct experiments only on the Medium subset.

By default, we use the same experimental setup across all experiments: LLaMA-3.1-8B is used as both the memory extraction (graph construction) model and the answer generation model, while Contriever (Izacard et al., 2021) is used as the embedding model. We use gpt-4o as the judge model for LongMemEval and use gpt-4o-mini as the judge model for HaluMem. During the QA experiments, for Value = session, we select the top-5 values to generate the answer, whereas for Value = Key, we select the top-20 values for answer generation. All experiments are conducted on a machine with a machine with 8 NVIDIA H100 80GB GPUs.

Key Design	LME-S		LME-M	
	R@5	R@10	R@5	R@10
`session`	0.9021	0.9714	0.7112	0.8043
`session,S,F,K`	**0.9379**	0.9690	0.7327	0.8521
`[session,S,F,K]`	0.9165	0.9666	0.7184	0.8210
`session,[S,F,K]`	**0.9379**	**0.9833**	**0.7685**	**0.8592**
`S,F,K`	**0.9117**	0.9618	0.6921	0.8091
`[S,F,K]`	0.9045	**0.9642**	**0.7064**	**0.8138**

Table 2: Retrieval performance on LongMemEval under different keys. S=summary, F=aggregated factual statements, K=aggregated keywords. `[a,b]` means their representations are merged during retrieval. R@n means Recall@n.

4.2 How to select Keys

As discussed in Section 3.1, existing works differ substantially in the representation forms they adopt in practice. The design of keys plays a critical role in memory systems. This involves two main questions: (1) what types of keys to extract, and (2) how to organize these keys into memory.

In this section, we analyze and evaluate three commonly used types of keys—*summaries*, *factual statements*, and *keywords*—together with two organization strategies, *merge-by-type* and *merge-by-session* [2]. For *merge-by-type*, the query is matched against each merged content to obtain similarity scores, which are averaged to produce the final score for the session.

As shown in Table 2, augmenting session-based keys (i.e., raw dialogue content) with three types of derived information at different granularities consistently improves retrieval performance, regardless of the organization strategy employed. In contrast, the merge-by-type organization (`session,S,F,K`) outperforms merge-by-

[2]We omit the separate strategy as it has already been shown to perform poorly in the LongMemEval paper. Interestingly, we find that the separate strategy achieves the best performance on the HaluMem dataset instead; details can be found in the Appendix C.

Key Design	op: add			op: add / update / noop			
	Mem-R	Mem-P	QA-C	Mem-R	Mem-P	MemUpdate-C	QA-C
`S,F,K`	0.7332	0.9360	0.2815	0.8069	0.8829	0.1416	0.5134
`[S,F,K]`	0.7332	0.9360	**0.4785**	0.8057	0.8846	**0.2207**	**0.5861**

Table 3: Performance on HaluMem-medium under different operation settings. "Mem-R" denotes Memory Extraction Recall, "Mem-P" denotes Target Memory Extraction Precision, MemUpdate-C denotes Correct Ratio of memory update operation, and QA-C denotes Correct Ratio of end-to-end Question Answering.

value (`[session,S,F,K]`), which differs from the conclusions reported in LongMemEval. Moreover, we find that merging only the derived information while keeping the raw content as an independent key (`session,[S,F,K]`) yields better performance than the standard merge-by-value strategy. For settings where raw content is not allowed as a key, the merge-by-value strategy shows a slight advantage.

Takeaway. If retaining the session is allowed, use `session,[S,F,K]` as keys organization, otherwise try (`[S,F,K]`) first.

4.3 How to select Memory Operations

As discussed in Section 3.3, a key distinction between dialog memory systems and standard RAG pipelines is that memory indexing often involves explicit maintenance operations beyond simple insertion. Although many existing memory systems support update or alignment mechanisms in principle, their reported end-to-end results are often outperformed by systems that only perform add operations[3]. As a result, the impact of different maintenance operations is rarely isolated or systematically analyzed.

In this section, we explicitly evaluate the effectiveness of *Update* and *Noop* operations on HaluMem, a benchmark designed to assess memory extraction and update. Following recent findings (Ong et al., 2025) that direct deletion is generally unnecessary in dialog memory systems, we omit the *Delete* operation from our analysis.

As shown in Table 3, introducing the *Update* and *Noop* operations leads to a decrease in memory precision, likely because these operations may incorrectly modify memories. In contrast, memory recall improves substantially, which ultimately translates into higher end-to-end QA accuracy. Comparing the *merge-by-type* and *merge-by-value* key organization strategies, we observe that latter consistently performs better, regardless of whether the *Update* and *Noop* operations are applied. This results are consistent with Table 2 and exhibit more pronounced differences, which may be attributed to the characteristics of the dataset.

[3]Details can be found in Table D.2.

Graph Design	LME-S		LME-M	
	R@5	R@10	R@5	R@10
`[S,F,K]`	0.9045	0.9642	0.7064	0.8138
SimGraph	0.8424	0.9498	0.6043	0.7766
KnowGraph	0.9116	0.9665	0.6610	0.7828
DescGraph	**0.9331**	**0.9713**	**0.7661**	**0.8735**

Table 4: Retrieval performance on LongMemEval under different graph schema.

Takeaway. The *Update* and *Noop* operations are indeed useful and are worth considering in practice.

4.4 How to construct graphs

As discussed in Section 3.3, the key to constructing a graph index lies in how nodes and edges (connections) are defined. For lightweight design, many existing works (Xu et al., 2025; Wu et al., 2025b) directly build upon an existing flat index, introducing edges based on textual similarity or membership relations. Another line of work (Jimenez Gutierrez et al., 2024; Chhikara et al., 2025) leverages LLMs to extract entities and semantic relations from raw content, thereby constructing a knowledge-graph-style memory index.

In this section, based on LongMemEval, we compare these two common graph construction paradigms and further propose a simple yet effective improvement.

SimGraph: Following A-mem, for each Key Group $[S_i, F_i, K_i]$, we retrieve the top-5 most similar Key Groups via vector matching and then employ an LLM to determine whether an edge should be established.

KnowGraph: A Knowledge Graph composed of triples $\langle subject, predicate, object \rangle$, where nodes (subjects/objects) correspond to entities and edges (predicates) represent relations between en-

Activation Design	R@5	R@10	N@5	N@10
Entities	**0.9642**	**0.9905**	**0.9599**	**0.9648**
Triples	0.9403	0.9809	0.9363	0.9460

Table 5: Graph Activation method comparison on LongMemEval-S. N@n means NDCG@n.

Re-rank Design	no expansion		one-hot expand	
	R@5	R@10	R@5	R@10
$Score_s$	0.9546	0.9904	0.8615	0.9618
$Score_e$	**0.9642**	0.9904	0.9594	0.9643
$(Score_e, Score_g)$	**0.9642**	0.9904	**0.9642**	**0.9928**

Table 6: Retrieval performance on LongMemEval under different ranking methods.

tities. A potential limitation of this construction paradigm is its restricted representational capacity. Similar to factual statements, triples are primarily suited for expressing semantic memory, but are less effective at capturing episodic memory. In flat indices, episodic memory is typically carried by summaries, which allow agents to describe and record events in a relatively free-form manner.

DescGraph: We adopt a direct approach by enriching entities with natural language descriptions, which record both semantic memory and episodic memory associated with each entity. During the construction of DescGraph, these descriptions are updated as nodes and edges are merged, in a manner analogous to the memory update operation in flat indices.

Retrieval Setup: For all graph variants, we employ the same retrieval strategy: Q $\rightarrow$ K $\rightarrow$ 1-Hop $\rightarrow$ V (first match initial nodes and then perform a one-hop expansion, values are selected by ranking nodes according to their query–node similarity scores.). For KnowGraph, node representations are derived from entity names, whereas for DescGraph, node representations are based on node descriptions.

As shown in Table 4, using SimGraph actually led to worse retrieval performance. The underlying reason is that graph expansion based on similarity edges introduces excessive noise without an effective reranking mechanism, making it less effective than using only the initially activated nodes. The experimental results demonstrate that DescGraph consistently achieves superior performance. More details can be found in Appendix E.

Takeaway. Constructing graphs based on similarity is hard to make effective, whereas adding entity descriptions on top of an entity–relation–based graph yields clear improvements.

4.5 How to search graphs

As introduced in Section 3.4.2, graph retrieval can be decomposed into two stages: activation and expansion. In this section, we compare two commonly used graph activation strategies:

- *Entity activation*: the query is matched against entity embeddings, and the top-k entities are selected.
- *Triple/Relation activation*: the query is matched against triple embeddings, and the top-k triples are selected.

We further evaluate one of the most commonly adopted graph expansion methods, 1-hop expansion: traverses each initially activated entity and add its 1-hop neighbors to the activated entity set.

Since the number of values (i.e. sessions) associated with the activated entity set may exceed the return budget of the system, a re-ranking step is required. We consider two straightforward scoring strategies:

- $Score_s$: the vector similarity between the query and the session (value).
- $Score_e$: the vector similarity between the query and the entity (key).

As a single entity may correspond to multiple sessions, relying solely on $Score_e$ often leads to score ties. We therefore introduce an additional secondary order key $Score_g$ denoting the number of candidate keys associated with a value, where candidate keys include the currently activated entities and their 1-hop neighbors.

The two methods reported in Table 5 do not apply graph expansion and use the same re-ranking strategy (S_e, S_g). The results show that direct entity activation consistently outperforms triple-based activation. Furthermore, Table 6 indicates that, under the augmented re-ranking configuration, the performance difference between enabling and disabling 1-hop expansion is marginal.

However, when only $Score_s$ is used, the performance of the expansion degrades noticeably. This is because 1-hop expansion introduces a larger set of candidate values, imposing a higher demand on re-ranking quality. After incorporating $Score_g$ as a secondary key to $Score_e$, the relative improvement under the 1-hop expansion setting becomes more pronounced, further supporting this explanation.

Extraction	Embedding	Answering	Index	Retrieval				Answer Accuracy	
				R@5	R@10	N@5	N@10	V=session	V=Key
LongMemEval-S									
llama-3.1-8b	contriever	llama-3.1-8b	flat	0.9045	0.9642	0.9100	0.9207	0.614	**0.570**
llama-3.1-8b	contriever	llama-3.1-8b	graph	**0.9356**	**0.9761**	**0.9393**	**0.9477**	**0.620**	0.518
gpt-4o-mini	text-emb-3-s	gpt-4o	flat	0.9284	0.9880	0.9346	0.9458	0.760	**0.752**
gpt-4o-mini	text-emb-3-s	gpt-4o	graph	**0.9690**	**0.9928**	**0.9711**	**0.9742**	**0.892**	0.690
LongMemEval-M									
llama-3.1-8b	contriever	llama-3.1-8b	flat	0.7064	0.8138	0.7309	0.7600	0.526	**0.478**
llama-3.1-8b	contriever	llama-3.1-8b	graph	**0.7661**	**0.8735**	**0.8138**	**0.8377**	**0.548**	0.428
gpt-4o-mini	text-emb-3-s	gpt-4o	flat	0.7231	0.8353	0.7292	0.7588	0.638	**0.620**
gpt-4o-mini	text-emb-3-s	gpt-4o	graph	**0.8281**	**0.9307**	**0.8631**	**0.8880**	**0.754**	0.592

Table 7: Comparison of two strong baselines on LongMemEval. flat: Key=[S,F,K], Op=Add. graph: Key=entity description, Retrieval=Q $\rightarrow$ K $\rightarrow$ V, Rank=$(Score_e, Score_g)$

Extraction	Embedding	Answering	Index	Extraction			Answer (V=Key)		
				Mem-R	Mem-P	MemUpdate-C	QA-C	QA-H	QA-O
llama-3.1-8b	contriever	llama-3.1-8b	flat	**0.8057**	0.8846	0.2207	**0.5861**	**0.1855**	**0.2284**
llama-3.1-8b	contriever	llama-3.1-8b	graph	0.4742	**0.9921**	-	0.4935	0.2307	0.2757
gpt-4o-mini	text-emb-3-s	gpt-4o	flat	**0.7759**	0.8809	0.2691	0.5645	**0.1578**	0.2778
gpt-4o-mini	text-emb-3-s	gpt-4o	graph	0.6493	**0.9936**	-	**0.6322**	0.2059	**0.1618**

Table 8: Comparison of two strong baselines on HaluMem-medium. flat: Key=[S,F,K], Op=Add/Update/Noop. graph: Key=entity description, Retrieval=Q $\rightarrow$ K $\rightarrow$ V, Rank=$(Score_e, Score_g)$

Takeaway. If Value=session, directly activating entities (w/o expansion) and rank value according to $(Score_e, Score_g)$ is a strong baseline.

4.6 End-to-end Results

Finally, we adopt a setting that disallows using raw content as keys and select one best-performing configuration for each of the flat based and graph based methods on both LongMemEval and HaluMem, and conduct a full comparison of their results. We primarily conducted end-to-end experiments under two settings: LLaMA- 3.1-8B with Contriever, and OpenAI GPT-4o-mini with text-embedding-3-smal. We also evaluated Qwen3-8B, and its results on LongMemEval-S are consistent with those of the two settings above. More results can be found in Appendix C.

LongMemEval. In terms of retrieval performance, the graph method consistently outperforms the flat index method across different model combinations. This gap becomes more pronounced as the dataset scale increases from S to M.

For question answering, under the Value = session setting, the graph method achieve better performance overall, with particularly notable improvements under the higher-capacity setting (using gpt-4o-mini for graph construction). This suggests that, compared to directly extracting flat key such as factual statements, *graph construction (extracting entities, relations, and descriptions) requires stronger model capabilities*.

Under the Value=Key setting, however, the graph method performs worse. This is because each key representation used in the flat index method (i.e., [S,F,K]) aggregates all derived information from a single session. In contrast, the information contained in entity descriptions is not sufficiently rich. Consequently, when the same number of keys is selected for answering, the flat index method effectively has access to substantially more information—essentially comparable to using sessions.

HaluMem. HaluMem does not include explicit evaluation of retrieval metrics; instead, it measure the memory extraction performance. The QA results on HaluMem are consistent with the conclusions drawn on LongMemEval: when using a weaker extraction model and setting V=key, the graph method performs poorly. Notably, graph method also exhibit lower recall in memory extraction metrics, which may stem from the granularity mismatch between the ground-truth memories (biased toward factual statements) and the memories represented in graphs (entity-centric descriptions).

5 Conclusion

This paper presents a systematic analysis of long-term dialog memory architectures. Motivated by inconsistent empirical findings in prior work, we propose a unified analytical framework that decomposes dialog memory systems into core components, enabling controlled comparison across graph-based and non-graph approaches.

We evaluate common design choices in dialog memory systems, Our results show that (1) **foundational system settings have a substantial impact on performance**; (2) **while graph-based memory can offer advantages under certain configurations, inappropriate graph construction or retrieval strategies may instead degrade results**.

Overall, this work provides a practical analytical framework and a set of strong baselines that help clarify the design space of dialog memory systems.

Limitations

First, the unified memory framework we propose is intended to cover the majority of commonly used memory systems, helping the community focus on shared underlying implementation details and facilitating fair comparisons. However, we acknowledge that the proposed framework cannot cover all possible memory systems. In practice, memory systems can be highly customized; for example, Memory OS treats memory as a schedulable system resource and manages it by partitioning memory into long-term, mid-term, and short-term components.

Second, to derive relatively general conclusions and a set of strong baselines, we conduct experimental analyses under several commonly adopted settings at different stages of the pipeline. We do not exhaustively explore all possible design choices, such as alternative forms of memory keys, heterogeneous or hierarchical graphs (Gutiérrez et al., 2025), more sophisticated graph retrieval methods (Zhuang et al., 2025; Luo et al., 2025; Hu et al., 2018a,b), or approaches that require additional training (Wang et al., 2025; Yan et al., 2025). Meanwhile, more fine-grained comparative experiments remain to be explored, such as disentangling and comparing the Update and Noop operations.

Third, our experiments are conducted on two widely recognized and complementary benchmarks, LongMemEval and HaluMem. Nevertheless, some of our conclusions may vary on other memory benchmarks with different characteristics. In fact, selecting different configurations for different datasets (and real-world application scenarios) is a reasonable and practical choice. Specifically, our experimental findings regarding the choice of [S, F, K] may not directly generalize to non-dialogue memory tasks, such as long-context RAG. In contrast, generic long-text RAG typically keeps the full original documents indexed and retrievable, making it less necessary to completely replace raw text with [S, F, K] representations.

Finally, considering both local deployment and API-based settings, we primarily evaluate only two backbone model configurations (LLaMA-3.1-8B with Contriever, and OpenAI GPT-4o-mini with text-embedding-3-small), along with partial experiments on Qwen3-8B, without exploring a broader range of combinations.

References

Ding Chen, Simin Niu, Kehang Li, Peng Liu, Xiangping Zheng, Bo Tang, Xinchi Li, Feiyu Xiong, and Zhiyu Li. 2025. Halumem: Evaluating hallucinations in memory systems of agents. *arXiv preprint arXiv:2511.03506*.

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

Xingjian Diao, Chunhui Zhang, Weiyi Wu, Zhongyu Ouyang, Peijun Qing, Ming Cheng, Soroush Vosoughi, and Jiang Gui. 2025. Temporal working memory: Query-guided segment refinement for enhanced multimodal understanding. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3393–3409.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z Pan. 2025. Rethinking memory in ai: Taxonomy, operations, topics, and future directions. *arXiv preprint arXiv:2505.00675*.

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

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, and 1 others. 2025. Lightmem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*.

Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2018a. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30(5):824–837.

Sen Hu, Lei Zou, and Xinbo Zhang. 2018b. A state-transition framework to answer complex questions over knowledge base. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2098–2108, Brussels, Belgium. Association for Computational Linguistics.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, and 1 others. 2025. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Wenqi Jiang, Shuai Zhang, Boran Han, Jie Wang, Bernie Wang, and Tim Kraska. 2024. Piperag: Fast retrieval-augmented generation via algorithm-system co-design. *arXiv preprint arXiv:2403.05676*.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569.

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

Rui Li, Zeyu Zhang, Xiaohe Bo, Zihang Tian, Xu Chen, Quanyu Dai, Zhenhua Dong, and Ruiming Tang. 2025. Cam: A constructivist view of agentic memory for llm-based reading comprehension. *arXiv preprint arXiv:2510.05520*.

Haoran Luo, Yikai Guo, Qika Lin, Xiaobao Wu, Xinyu Mu, Wenhao Liu, Meina Song, Yifan Zhu, Luu Anh Tuan, and 1 others. 2025. Kbqa-o1: Agentic knowledge base question answering with monte carlo tree search. *arXiv preprint arXiv:2501.18922*.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4089–4100.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. 2025. Nemori: Self-organizing agent memory inspired by cognitive science. *arXiv preprint arXiv:2508.03341*.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seungwon Hwang, Dongha Lee, and Jinyoung Yeo. 2025. Towards lifelong dialogue agents via timeline-based memory management. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8631–8661.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Peijun Qing, Xingjian Diao, Chiyu Ma, Saeed Hassanpour, and Soroush Vosoughi. 2026. Tailoring memory granularity for multi-hop reasoning over long contexts. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 3648–3666.

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

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, and 1 others. 2025. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8416–8439.

Zhenheng Tang, Xin He, Tiancheng Zhao, Fanjunduo Wei, Xiang Liu, Peijie Dong, Qian Wang, Qi Li, Huacan Wang, Ronghao Chen, and 1 others. 2026. Llm agent memory: A survey from a unified representation–management perspective.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. 2025. Mem-{\alpha}: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025a. From human memory to ai memory: A survey on memory mechanisms in the era of llms. *arXiv preprint arXiv:2504.15965*.

Yaxiong Wu, Yongyue Zhang, Sheng Liang, and Yong Liu. 2025b. Sgmem: Sentence graph memory for long-term conversational agents. *arXiv preprint arXiv:2509.21212*.

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

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

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu Chainof-note. 2023. Enhancing robustness in retrieval-augmented language models, 2023. *URL https://arxiv.org/abs/2311.09210*.

Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, and 1 others. 2025a. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958*.

Yingyi Zhang, Pengyue Jia, Derong Xu, Yi Wen, Xianneng Li, Yichao Wang, Wenlin Zhang, Xiaopeng Li, Weinan Gan, Huifeng Guo, and 1 others. 2025b. Personalize before retrieve: Llm-based personalized query expansion for user-centric retrieval. *arXiv preprint arXiv:2510.08935*.

Yujie Zhang, Weikang Yuan, and Zhuoren Jiang. 2025c. Bridging intuitive associations and deliberate recall: Empowering llm personal assistant with graph-structured long-term memory. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 17533–17547.

Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. 2025. Linearrag: Linear graph retrieval augmented generation on large-scale corpora. *arXiv preprint arXiv:2510.10114*.

Appendices

A Unified Framework Details

A.1 Forms of Keys

Common forms of Keys include:

- **Summaries:** typically in session-level;
- **Statements:** natural language assertions, suitable for factual content or event description;
- **Keywords:** key phrases extracted from the original text;
- **Tags/Categories:** abstract labels for grouping or classification;
- **Entities:** persons, organizations, products, concepts, etc.;
- **Triples:** $\langle subject, predicate, object \rangle$, typically used with graph structures.

A.2 Semantic or Episodic Memory

Inspired by human memory, recent work tends to categorize dialogue memory into:

- **Semantic Memory:** stable facts, attributes, and relationships; often represented as statements or triples (e.g., "the user likes sugar-free drinks" or $\langle$User, likes, sugar-free drinks$\rangle$);
- **Episodic Memory:** event-like records of "who did what, where, and when"; represented as summaries or statements with time, location, and participating entities (e.g., "On 2024-11-10 in Shanghai, the user asked about EU privacy-law compliance.").

A.3 Memory Index Operations

- **Add**: Newly extracted memories are appended to the index as new entries.
- **Update**: Certain existing memories are updated or revised based on newly extracted ones.
- **Delete**: Some existing memories are removed from the index.
- **Noop**: Newly extracted memories are discarded and not added to the index.

Recent work (Ong et al., 2025) argues that directly deleting old memories is undesirable, as they may still be useful in the future (e.g., when a user refers to a former occupation). When a user query only concerns the most recent information, outdated memories can instead be filtered out during retrieval using timestamps.

A.4 Graph Index and Retrieval

Graph indices connect keys via explicit edges.

Edge Types Common edge types include:

- **Similarity edges:** connect semantically similar keys according to embedding distance;
- **Part-of edges:** represent that a key belongs to a higher-level node;
- **KG-style relation edges:** as in HippoRAG2 and Mem0, edges are labeled with predicates between entities.
- **Temporal edges:** capture temporal order (e.g., adjacency within the same session);
- **Causal edges:** represent causal relationships where one event leads to another;

Node Types Different systems instantiate graph nodes in different ways:

- **Text chunks as nodes:** e.g., CAM, where chunk-level text segments are nodes;
- **Sentences as nodes:** e.g., SG-mem, using sentence-level nodes for finer granularity and more precise selection;
- **Entities as nodes:** e.g., GraphRAG, Zep, Mem0:
- **Hybrid nodes:** e.g., HippoRAG2 uses a hybrid scheme that can be viewed as hierarchical:

Hierarchical Graphs Hierarchical graphs connect multiple semantic levels: Some prior work, such as SGMem, treats the inherent subordinate relationship between keys and values as "edges," thereby constructing a so-called "hierarchical graph." In contrast, in our framework this relationship is regarded as a fixed mapping. Even a flat index can naturally preserve such a mapping, and therefore it is not considered a hierarchical graph.

Correspondingly, more representative examples of true hierarchical graphs include the Insight–Query–Interaction hierarchy in G-Memory and the Community–Entity hierarchical structure in GraphRAG. Since the former is primarily designed for system memory, this paper focuses its discussion on the latter.

Graph Expansion Starting from the seed nodes, expand over the graph to gather additional context:

- **Simple neighbor expansion (BFS-style):** take h-hop neighbors (often $h = 1$) and truncate to a maximum number of nodes $N_{\max}$ when necessary;
- **Structure-aware expansion (HippoRAG2):** run PageRank or related algorithms from the seed nodes, exploiting graph topology; node

embeddings are used mainly to re-rank the expanded set;

- **Semantic-aware expansion (EcphoryRAG):** Compute a "center vector" from the seed nodes, and teratively expand by selecting top-k nodes semantically close to the center vector. Finally re-score all candidates against the query embedding to avoid semantic drift.

B Dataset Details

LongMemEval (Wu et al., 2024) primarily focuses on memory retrieval and reasoning over extremely long dialog histories and consists of two subsets (S and M). The benchmark covers a diverse set of task types, including information extraction, cross-session reasoning, temporal reasoning, knowledge updating, and refusal detection, making it well suited for evaluating memory organization and retrieval quality at scale.

HaluMem (Chen et al., 2025) is designed to evaluate memory extraction, updating, and consistency. HaluMem dialogs contain a large number of information updates and explicitly test whether memory systems can correctly extract new information, update outdated memories, and avoid hallucinated responses. Different from LongMemEval, HaluMem does not include retrieval-based evaluation metrics. It consists of Medium and Long subsets; due to the high computational cost, we conduct experiments only on the Medium subset.

It should be noted that all HaluMem evaluation metrics are based on LLM-as-Judge. In contrast, LongMemEval provides a ground-truth list of relevant sessions for each test query and computes retrieval metrics such as recall and NDCG based on this list. Considering both metric robustness and experimental efficiency, we primarily base our method comparisons and selection across different stages on the LongMemEval dataset.

C Extended Experimental Results

To supplement the core conclusions, this appendix offers an extended report of our experimental results from Section 4.6. Furthermore, we include ancillary analysis and exploratory comparisons to discuss the trade-offs and empirical motivations behind several minor design configurations.

C.1 Full Evaluation Metrics

Figure C.1 presents the Question Answering (QA) and retrieval performance across our four primary experimental configurations on the $\mathrm{LongMemEval_S}$ and $\mathrm{LongMemEval_M}$ benchmarks. The results are visualized using radar charts to provide a comprehensive assessment of model capabilities across six distinct problem dimensions: Single Session User, Single Session Preference, Knowledge Update, Single Session Assistant, Temporal Reasoning, and Multi-Session.

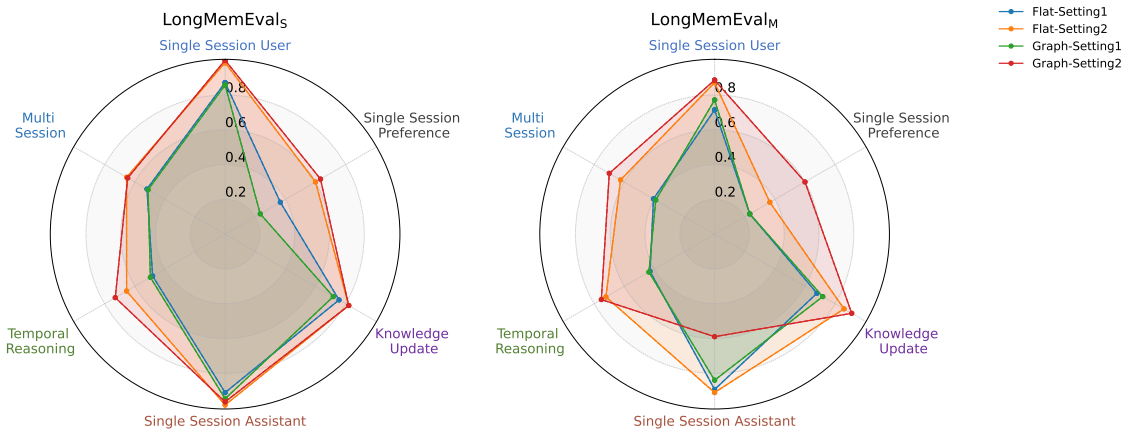
The evaluation compares four experimental settings, which are consistent with the configurations detailed in Table 7. In this framework, "Flat" and "Graph" denote the specific indexing methodologies employed for memory organization. The term "Setting" distinguishes the scale and capacity of the Large Language Models (LLMs) used for memory construction and retrieval.

- **Setting 1 (Local Deployment)**: Utilizes Llama-3.1-8B for memory extraction and QA, paired with Contriever for generating embeddings. This configuration represents a more constrained experimental environment.
- **Setting 2 (API Service)**: Employs GPT-4o-mini for extraction and QA, alongside text-embedding-3-small for embedding tasks.

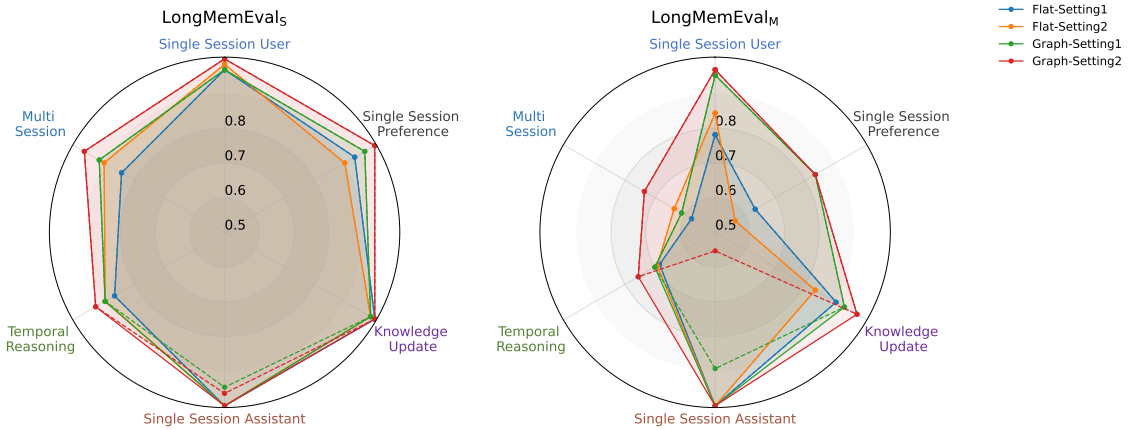
For all experiments, GPT-4o is utilized as the backbone evaluator to ensure consistency. As Setting 1 operates under lower specifications, its performance is generally expected to be inferior to that of Setting 2.

Interestingly, Graph-setting2 exhibits an unexpected performance drop in Single Session Assistant queries — a phenomenon also noted but left unanalyzed in the Zep and Nemori (Rasmussen et al., 2025; Nan et al., 2025). We identify the cause as a mismatch in information sourcing: while the ground truth for these questions lies in the assistant's responses, standard efficiency-driven practices—including ours—only use user messages to build the graph. Consequently, the graph itself lacks the necessary answer nodes. Our prejudge mechanism exacerbates this by filtering out messages it deems "uninformative," effectively removing the bridge between the question and the answer. Counter-intuitively, this issue becomes more severe with larger models, as their higher precision leads to stricter (and thus more destructive) filtering. This presents a clear trade-off: the prejudge mechanism is essential for reducing overhead and boosting overall system recall, but it costs us performance in this specific sub-type. To maintain global efficiency, we ultimately retained this approach.

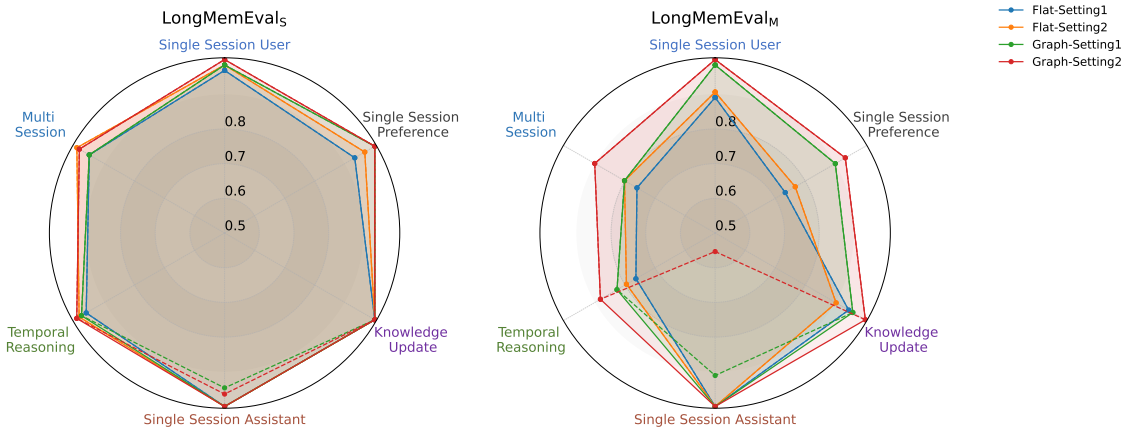
Table C.2 presents the complete experimental



(a) Answer Accuracy



(b) Retrieval Recall@5



(c) Retrieval Recall@10

Figure C.1: **Performance of Different Configurations Across Question Types in LongMemEval.** The plots illustrate results under two experimental environments: (1) **Setting 1 (Local Deployment)**, utilizing Llama-3.1-8B for extraction/QA and Contriever for embeddings; and (2) **Setting 2 (API Service)**, employing GPT-4o-mini and text-embedding-3-small. Dashed lines indicate retrieval metrics for Single-Session Assistant questions without filtering. In contrast, following the standard LongMemEval protocol, the primary values reported in our main text exclude questions where the ground truth resides outside user turns.

results on the HaluMem dataset, serving as a supplement to Table 8 in the main text. We observe that the strong baselines we propose consistently outperform the results reported in Table C.2, par-

ticularly on the Memory Recall metric. Since the HaluMem paper does not report the detailed experimental settings of each method (e.g., the choice of memory extraction models and embedding models), it is difficult to attribute the observed performance differences.

C.2 Experiments with Qwen model

As shown in Table C.4, we conducted additional experiments using the Qwen3-8B model on LongMemEval-S. The model was deployed locally with vLLM, and the corresponding $qwen3\_nonthinking.jinja$ template was used to disable the model's thinking mode. In these experiments, Qwen3-8B served as the LLM for both the extraction and question-answering stages, while Contriever was used as the retriever. The evaluation was performed using GPT-4o. The experimental results on the LongMemEval-S dataset show that Qwen3-8B slightly outperforms Llama-3.1-8B overall. Meanwhile, the relative performance trends between the flat and graph configurations remained largely unchanged.

C.3 Cost-Efficiency Analysis

Table C.5 summarizes the computational cost of flat and graph-based dialog memory systems under practical implementation settings. Statistics are reported over *unique* dialog sessions and tokens, as memory extraction is cached across repeated content. All experiments use `gpt-4o-mini` for memory extraction, `text-embedding-3-small` for embedding, and multi-threaded execution with `max_workers=16`.

Retrieval latency. Retrieval latency directly affects user experience and is therefore the most critical efficiency metric. Graph-based retrieval is consistently slower than flat retrieval, but remains within a practically acceptable range. On LongMemEval-S, both methods exhibit comparable latency (45 vs. 44 ms per query). On LongMemEval-M, graph retrieval incurs 574 ms per query compared to 240 ms for flat memory. Even at this scale (over 50k unique sessions), retrieval latency remains on the order of a few hundred milliseconds, which is typically acceptable for interactive dialog systems.

Memory extraction cost. Memory extraction is usually not on the critical path of user interaction and is often performed asynchronously after a session ends. Graph-based memory construction is more expensive than flat memory due to additional processing such as entity extraction and description aggregation. However, when normalized by the number of unique sessions, the cost remains manageable: on LongMemEval-M, graph extraction takes about 2.1 seconds per session, compared to 0.5 seconds for flat memory.

Memory storage. Both flat and graph-based memory achieve substantial compression relative to raw dialog content. Although graph memory stores more tokens than flat memory, the difference is modest compared to the original dialog size. For example, on LongMemEval-M, over 100M tokens of dialog are compressed into fewer than 30M tokens of stored memory in both settings.

Summary. Flat memory offers lower construction and retrieval cost, while graph-based memory introduces additional overhead that increases with scale. Nevertheless, this overhead remains manageable under realistic usage patterns, suggesting that efficiency alone should not preclude the adoption of graph-based designs when they provide modeling benefits.

C.4 Supplementary Design Experiments

In this subsection, we delve into the empirical rationale behind several design choices. By comparing alternative configurations, we justify the specific settings adopted in our final framework.

C.4.1 Key Selection on HaluMem

Table C.6 shows the results of key design experiments conducted on HaluMem. To perform the retrieval recall analysis on HaluMem, we adopted the evaluation protocol of LongMemEval. Since HaluMem does not natively provide the source session for each question's evidence, we constructed the ground truth by matching the evidence with the memory points across all sessions.

As shown in Table C.6, the results on HaluMem remain consistent with our findings in Table 2. The introduction of refined keys via the separate strategy (session, S, F, K) yields superior results in top-5 retrieval recall. Conversely, the merge strategy often leads to performance degradation, with [session, S, F, K] even underperforming the session-only baseline.

When raw session content is unavailable, the separate strategy (S, F, K) significantly outperforms the merge strategy ([S, F, K]). This confirms that

Model Setting	Key Org	Operation	Edge	Extraction			Answer (V=Key)		
				Mem-R	Mem-P	MemUpdate-C	QA-C	QA-H	QA-O
Local Deploy	`s,f,k`	A	N	0.7332	0.9360	-	0.5492	0.1263	0.3245
API service	`s,f,k`	A	N	0.7686	0.8926	-	0.6637	0.1526	0.1837
Local Deploy	`session,S,F,K`	A/U/N	N	0.7121	0.9811	0.1095	0.4800	0.2157	0.3043
Local Deploy	`[session,S,F,K]`	A/U/N	N	0.7124	0.9808	0.1537	0.5535	0.1552	0.2913
Local Deploy	`[session,S,F,K]`	A/U/N	Y	0.7124	0.9808	0.1537	0.5524	0.1635	0.2841

Table C.1: External flat-based results on HaluMem-medium. Key=`s,f,k` refers to Separate organization; Key=`session,S,F,K` refers to Merge-by-type organization; Key=`[session,S,F,K]` refers to Merge-by-all organization. A=Add, U=Update, N=Noop.

Dataset	**System**	Memory Integrity		Memory Accuracy			Memory Updating			Question Answering		
		R	Weighted R	Target P	Acc.	FMR	C	H	O	C	H	O
Medium	Mem0	42.91	65.03	86.26(10556)	60.86(16291)	56.80	25.50	0.45	74.02	53.02	**19.17**	27.81
	Mem0-Graph	**43.28**	**65.52**	87.20(10567)	**61.86(16230)**	55.70	24.50	0.26	75.24	54.66	19.28	26.06
	Memobase	14.55	25.88	**92.24(5443)**	32.29(17081)	**80.78**	5.20	0.55	94.25	35.33	29.97	34.71
	Supermemory	41.53	64.76	90.32(14134)	60.83(22551)	51.77	16.37	1.15	82.47	54.07	22.24	23.69
	Zep	-	-	-	-	-	**47.28**	0.42	**52.31**	**55.47**	21.92	**22.62**
Long	Mem0	3.23	11.89	88.01(1134)	**46.01(2433)**	87.65	1.45	0.03	98.51	28.11	**17.29**	54.60
	Mem0-Graph	2.24	10.76	87.32(785)	41.26(1866)	**88.36**	1.47	0.04	98.40	32.44	21.82	45.74
	Memobase	6.18	14.68	**88.56(3077)**	25.61(11795)	85.39	4.10	0.36	95.38	33.60	29.46	36.96
	Supermemory	**53.02**	**70.73**	85.82(24483)	29.71(77134)	36.86	17.01	0.58	82.42	**53.77**	22.21	**24.02**
	Zep	-	-	-	-	-	**37.35**	0.48	**62.14**	50.19	22.51	27.30

Table C.2: Original evaluation results reported in HaluMem paper.

maintaining independent keys is more robust, especially under constrained retrieval settings.

C.4.2 Ablation Study on the Add Operation

In Section 4.3, we primarily investigated two memory update strategies: *add* and *add/update/noop*. To further evaluate the specific contribution of the `add` operation to the overall system performance, we conducted an ablation study as presented in Table C.7.

In this experiment, we compare two settings under the Separate(S, F, K) indexing strategy:

- **w/ add**: A comprehensive update strategy including `add`, `update`, and `noop`.
- **w/o add**: A restricted strategy that only allows `update` and `noop`.

The results indicate that while memory extraction metrics remain relatively stable, the inclusion of the `add` operation significantly enhances memory update accuracy and final end-to-end QA performance. This underscores the necessity of the `add` operation for maintaining an up-to-date and comprehensive memory index. Consequently, the "w/ add" configuration demonstrates superior overall performance; thus, we adopt this setting for all update operations discussed in the main text.

C.4.3 Ablation on Prejudge Mechanism

The Prejudge Mechanism is a strategic filtering step integrated into our memory construction pipeline. For each incoming text chunk, the system employs an LLM to evaluate its informativity before performing fine-grained extraction. Chunks deemed irrelevant or redundant are immediately discarded, while only those identified as valuable proceed to the subsequent information extraction and graph update stages.

As shown in Table C.8, we compare the performance with the Prejudge Mechanism enabled (`w/`) and disabled (`w/o`). The results demonstrate that incorporating Prejudge does not compromise retrieval or ranking performance; in fact, it yields slight improvements across all metrics. By filtering out noise at the source, the mechanism effectively maintains a high signal-to-noise ratio in the memory index while substantially reducing unnecessary computational overhead for the subsequent update and storage modules. Consequently, the Prejudge Mechanism is employed in all experimental configurations throughout this paper unless otherwise noted.

Model Setting	Key Org	Retrieval		Answer Accuracy	
		R@5	R@10	V=session	V=Key
LongMemEval-S					
Local Deploy	`session,[S,F,K]`	0.9379	0.9833	0.638	0.594
API service	`session,[S,F,K]`	0.9498	0.9904	0.756	0.774
LongMemEval-M					
Local Deploy	`session,[S,F,K]`	0.7685	0.8592	0.562	0.496
API service	`session,[S,F,K]`	0.7661	0.8592	0.660	0.660

Table C.3: External flat-based results on LongMemEval. Key=`session,[S,F,K]` refers to refined Merge-by-session organization.

Extraction	Embedding	Answering	Index	Retrieval				Answer Accuracy	
				R@5	R@10	N@5	N@10	V=session	V=Key
LongMemEval-S									
llama-3.1-8b	contriever	llama-3.1-8b	flat	0.9045	0.9642	0.9100	0.9207	0.614	**0.570**
llama-3.1-8b	contriever	llama-3.1-8b	graph	**0.9356**	**0.9761**	**0.9393**	**0.9477**	**0.620**	0.518
qwen3-8b	contriever	qwen3-8b	flat	0.9069	0.9570	0.9227	0.9323	0.618	**0.600**
qwen3-8b	contriever	qwen3-8b	graph	**0.9427**	**0.9618**	**0.9495**	**0.9540**	**0.708**	0.576
gpt-4o-mini	text-emb-3-s	gpt-4o	flat	0.9284	0.9880	0.9346	0.9458	0.760	**0.752**
gpt-4o-mini	text-emb-3-s	gpt-4o	graph	**0.9690**	**0.9928**	**0.9711**	**0.9742**	**0.892**	0.690

Table C.4: Comparison of qwen model with other baseline on LongMemEval. flat: Key=`[S,F,K]`, Op=Add. graph: Key=entity description, Retrieval=Q $\rightarrow$ K $\rightarrow$ V, Rank=$(Score_e, Score_g)$

D Comparison with Related Works

D.1 Breakdown of Related Systems

Table D.1 decomposes and compares several representative dialog memory systems under a unified framework, the abbreviations and symbols used in the column headers and entries are explained below:

- **Key, Value:** Summ: Summary, Kw: Keyword, Ins: Insight, Comm: Community.
- **Query:** Q: Query, Q+T: Query + Time, Rew Q: Rewrited Query
- **Index Structure:** F: Flat, G: Graph, HG: Hybrid Graph, FG: Flat+Graph.
- **Index Operations:** Add: Add, Upd: Update, Align: Alignment, Comm Upd: Community Update, N/E Align: Node/Edge Alignment.
- **Retrieval:** $\rightarrow$: process flow, 1-Hop: one-hop neighbor retrieval, Rerank: re-ranking, PPR: Personalized PageRank, Kw: Keyword.
- **Answering:** CoN: Chain-of-note Answering, Direct: Direct Answering.

Interpretation Note: The table D.1 decomposes and compares the design choices across systems under our proposed unified framework. The *Key*, *Value*, and *Index* columns reveal how memory is organized. The *Retrieval* column depicts the search path complexity, and the *Query Answering* column distinguishes response generation methods.

D.2 Comparison of System Settings

Table D.2 summarizes the experimental settings and reported results of several representative dialog memory and RAG frameworks. Our analysis reveals two primary observations:

1. Misalignment in Evaluation Environments. There is a significant lack of uniformity in the foundational components used across different studies. As shown in the table, existing methods vary widely in their choice of information extraction models (ranging from Llama-3.1 to Gemini-1.5-Flash), embedding models (e.g., Stella V5, BGE-m3, text-embedding-3), and top-k retrieval constraints. Furthermore, the Value Type stored in memory—varying from raw Sessions and Facts to specific Graph Nodes or Episodic/Semantic units—further complicates direct performance comparisons. Such discrepancies in "system-level" configurations make it difficult to isolate the true effectiveness of any specific memory architecture or retrieval paradigm.

2. Performance Superiority via Systemic Refinement. Despite these inconsistent benchmarks, our framework achieves state-of-the-art performance across all metrics. Notably, beyond standard re-

Dataset	U Session #	U Session tok.	Method	Memory #	Memory tok.	Extr. (min)	Retr. (ms/q)
LME-s	19.2k	41.6m	flat	kw: 375.9k fact: 111.8k sum: 19.2k	kw: 1.3m fact: 1.9m sum: 2.0m	156	45
			graph	node: 156.1k edge: 145.9k	desc: 3.6m	1181	44
LME-m	51.6k	107.6m	flat	kw: 1,042k fact: 295.1k sum: 51.6k	kw: 3.5m fact: 5.0m sum: 5.5m	452	240
			graph	node: 1,218k edge: 1,266k	desc: 27.6m	1781	574
HaluMem-m	1387	3.2m	flat	kw: 32.7k fact: 19.3k sum: 1387	kw: 101k fact: 405k sum: 146k	308	15
			graph	node: 14.8k edge: 14.6k	desc: 438k	673	93

Table C.5: Statistics and efficiency of flat and graph memory construction

Key Design	R@5	R@10
`session`	0.4739	0.6002
`session,S,F,K`	0.4860	0.5988
`[session,S,F,K]`	0.4716	0.5994
`session,[S,F,K]`	0.4742	0.5924
`S,F,K`	0.4609	0.5795
`[S,F,K]`	0.4211	0.5160

Table C.6: Retrieval performance on HaluMem under different keys. S=summary, F=factual statement, K=keyword. `[a,b]` means their representations are merged during retrieval. R@n means Recall@n.

Update Op	Mem-R	Mem-P	MemUpdate-C	QA-C
`w add`	0.7121	0.9811	0.1095	0.4800
`w/o add`	0.7185	0.9814	0.0332	0.3754

Table C.7: Ablation results of the add operation on the HaluMem-medium dataset. "w/ add" denotes the full operation set (*add/update/noop*), while "w/o add" refers to the restricted set (*update/noop*). *Mem-R*, *Mem-P*, *MemUpdate-C*, and *QA-C* represent Memory Extraction Recall, Target Memory Extraction Precision, Correct Ratio of memory update operations, and end-to-end QA Accuracy, respectively.

Configuration	R@5	R@10	N@5	N@10
`w/o Prejudge`	0.9284	0.9689	0.9308	0.9386
`w/ Prejudge`	**0.9355**	**0.9761**	**0.9392**	**0.9476**

Table C.8: Ablation results of the Prejudge Mechanism. R@k and N@k denote Recall@k and nDCG@k, respectively. The results indicate that the Prejudge Mechanism optimizes computational efficiency without compromising (and even slightly improving) retrieval performance.

trieval recall (0.969 on LME-S), we emphasize our end-to-end QA accuracy, which reaches 0.892 and 0.754 using gpt-4o-mini on the S and M subsets, respectively. These results significantly surpass contemporary baselines, demonstrating that our gains are not merely a byproduct of backbone selection but a result of our systematic deconstruction of the dialog memory pipeline. By meticulously optimizing each modular component—from refined key design to update logic—our engineering-focused implementation proves that a unified and well-calibrated framework can overcome the inherent complexities of long-term memory management.

In summary, the substantial performance margins reported in Table 7 underscore the effectiveness of our design choices, providing a robust and reproducible benchmark for future research in long-term dialog memory systems.

E Case Study

This section presents several case studies.

E.1 Case study: Graph Index Failure Cases

As illustrated in Table 7, while the graph-based index achieves superior retrieval performance, this advantage does not consistently translate into downstream QA accuracy. Specifically, in the V = Session setting, the graph method performs on par with the flat baseline, whereas in the V = Key setting, it even underperforms. To investigate the underlying causes of this discrepancy, we conduct a qualitative analysis of representative cases where

	Key	Value	Query	Index (Struct; Op)	Retrieval	Answering
LongMemEval	Session + Fact	Session	Q + T	F; Add	Q → K → V	CoN
RMM	Topic summ	Session + Key	Q	F; Add, Upd	Q → K → V → Rerank	Direct
A-Mem	Session + Kw, Tag, Summ	Key	Q	G; Add, Upd	Q → K → V	Direct
Mem0-G	Entity name, Triple	Triple	Q	G; Add, Align	Q → K → 1-Hop → V	Direct
Zep	Entity summ, Triple, Comm	Key	Q	HG; Add, Align, Comm Upd	Q → K → 1-Hop → V → Rerank	Direct
SGMem	Sentence, Summ, Ins, Fact	Key	Q	FG; Add	Q → K → 1-Hop → V	Direct
Nemori	Episode+Semantic Memory	Key	Q	F; Add, Upd	Q → K → V	Direct
CAM	Chunk, Comm	Key	Q	HG; Add, Comm Upd	Q → K → 1-Hop → V	Direct
LightRAG	Entity name, Triple	Chunk + Key	Q → Kw	G; Add, Align,	Q → Kw → Entity+Triple→1-Hop → V	Direct
EcphoryRAG	Entity, Chunk	Key	Q	G; Add	Q → Entity+Chunk → 1-Hop → V	Direct
HippoRAG2	Entity, Triple, Chunk	Chunk	Rew Q	G; Add, N Align	Q → Triple → Node →PPR Rerank→ V	Direct
AriGraph	Triple	Chunk	Q	G; Add, E Align	Q → Triple → Node → 1-Hop →Triple→ V	Direct

Table D.1: Decomposition of some representative dialog memory systems under the unified framework.

							LongMemEval-S		LongMemEval-M	
Model	**Extraction**	**Embedding**	**QA Model**	**top-k**	**Value Type**	**Eval.**	**Recall**	**QA Acc.**	**Recall**	**QA Acc.**
LongMemEval	Llama-3.1-8B	Stella V5	gpt-4o	5/10	Session	gpt-4o	–	–	0.732/0.862	0.714/0.700
RMM	Gem-1.5-Flash	GTE	Gem-1.5-Flash	5/10	Session	Gem-1.5-Pro	–	–	0.698/0.744	0.704/0.738
A-mem	–	–	–	–	–	–	0.626/0.652*	–	–	–
Mem0-g	–	–	–	–	–	–	0.536/0.395*	–	–	–
Zep	gpt-4o-mini	BGE-m3	4o-mini/4o	20	Fact+Node	gpt-4o	0.638/0.712	–	–	–
SG-mem	Qwen2.5-32B	S-BERT	Qwen2.5-32B	5/10	Session	Qwen2.5-32B	–	–	–	0.700/0.730
Nemori	gpt-4o-mini	text-emb-3	gpt-4o-mini	10+20	Epi+Sem	gpt-4o-mini	–	0.744/0.794	–	–
LightMem	–	–	4o-mini/Qwen3	–	–	–	–	0.686/0.702	–	–
Ours	gpt-4o-mini	text-emb-3	gpt-4o-mini	5	Session	gpt-4o	**0.969/0.935**	**0.892/0.620**	**0.828/0.766**	**0.754/0.548**

Notes: **4o-mini/4o**: gpt-4o-mini/gpt-4o; **Qwen3**: Qwen3-30B-A3B-Instruct; **Gem-1.5-Flash/Pro**: Gemini-1.5-Flash/Pro; **text-emb-3**: text-embedding-3-small; **BGE-m3**: BAAI General Embedding; **S-BERT**: Sentence-BERT(all-MiniLM-L6-v2); **GTE**: General Text Embeddings; **Epi+Sem**: Episodic + Semantic Memory.Results marked with* are those reported in the LingtMem (Fang et al., 2025) paper.

Table D.2: Comparison of various dialog memory systems and RAG frameworks. results are collected from original papers/reports.

the graph strategy fails despite successful retrieval.

Figure E.1 presents a typical failure in the V = Session configuration. In this instance, the graph-based index retrieves and presents more time-related metadata within the prompt context. Analysis shows that for a smaller-scale model like Llama-3.1-8B, these additional temporal details act as distracting information. The model is indeed capable of recognizing the target fact (e.g., the specific internet speed of 500 Mbps) from the text; however, its reasoning process is misled by the salient yet irrelevant temporal noise, causing it to fail in selecting and outputting the correct answer. In contrast, the flat index strategy adopts a more concise organization of index, presenting context in a direct, narrative-driven manner. This approach not only enhances the expression of semantic information but also significantly reduces interference from irrelevant timestamp details, thereby retrieving the most semantic relevant information. This enables the model to maintain focus on the semantic core and successfully output the correct answer.

Figure E.2 reveals a failure mechanism in the V = Key setting. Here, the graph-based index retrieves 20 atomic entities (e.g., "TRIPLE THE PRICE", "FLEA MARKET"). Although relevant, this highly fragmented representation strips away essential descriptive context and semantic associations between entities. Consequently, the model fails to perform entity alignment—for example, it cannot confidently link the "painting of a sunset" in the query to the "flea market find" in the retrieved facts.In contrast, the flat strategy generates semantically cohesive "memory notes" by integrating key facts with multi-faceted summaries. Thus, the context provided by 20 memory notes carries significantly more relational information than 20 isolated entities. This structured representation preserves sufficient narrative anchors and semantic cohesion, enabling the model to bridge the semantic gap between the query and the evidence, even in the presence of price-related distractors.

Through these case studies, we attribute the decline in QA performance to two key factors. First, for capacity-limited models, organizing the retrieved context in a semantically coherent and structurally clear manner is crucial for downstream QA performance. Second, while the atomic entities and their concise descriptions used by the graph-based index contribute to its interpretability, in practice, this very design often results in informationally sparse representations, creating a

Case Study: Graph fail case(ID: ad7109d1)

User Query: *"What speed is my new internet plan?"*
Answer: 500 Mbps
Ground Truth Session: "user":I'm having some issues with my laptop's battery life. Can you give me some tips on how to extend it? "assistant":... "user":I did notice that my internet speed has been really good lately, especially when I'm streaming movies on Netflix. I upgraded to **500 Mbps** about three weeks ago, and it's made a huge difference. Anyway, back to my battery issue... do you think it's possible that my laptop's battery is just old and needs to be replaced? ...

Graph Index	Flat index
Retrieved Sessions: • **Session 1 (Target):** *...I upgraded to 500 Mbps about three weeks ago...* • **Session 2-3:** [Irrelevant: Maintenance schedules and shopping logs with heavy timestamps.] • **Session 4-5:** [Irrelevant: Distant historical logs regarding travel and sports.]	**Retrieved Sessions:** • **Session 2 (Target):** *...I upgraded to 500 Mbps about three weeks ago...* • **Session 1, 3-5:** [Irrelevant: Miscellaneous topics including yoga, meal prep, and shuttle services.]
Model Response (Hallucination): *"To answer your question...The history mentions you upgraded to 500 Mbps,* ***but it doesn't mention your current plan****...Can you please provide more context or information..."*	**Model Response (Success):** *"To answer your question...In Session 2, the user mentions that they upgraded to 500 Mbps about three weeks ago,...****The user explicitly mentions that they upgraded to 500 Mbps****, which is the speed of their new internet plan.Answer: Your new internet plan is 500 Mbps."*

Figure E.1: **A representative case where the flat index succeeds while the graph index fails.** In this case, because the graph-based index retrieves more time-related information, such content introduces distracting information into the prompt context, ultimately causing the less capable Llama-3.1-8B model to fail in providing the correct answer. In contrast, the higher information density of the flat index provides a richer semantic context, allowing the model to focus on the core information without being distracted by fragmented metadata."

performance bottleneck that must be carefully considered.

E.2 Case study: Graph Index Success Cases

To provide a comprehensive view, we further examine representative success cases of the graph index strategy in Figure E.3 and Figure E.4, both evaluated using the Llama3.1-8B model on the LongMemEval-S and LongMemEval-M benchmarks, respectively.

The phenomenon in Figure E.3 is exactly the opposite of that in Figure E.1. In this case, although both the graph index method and the flat index method retrieve the required session in the end, the noise from the retrieved chunks in the flat method interferes with generating the correct final result. In contrast, the session retrieved by the graph index method has less interference.

In Figure E.4, we observe a clear advantage of the graph index when handling complex, multi-session data. In this scenario, the high retrieval recall facilitated by the graph's topological awareness successfully translates into accurate QA results. Conversely, the flat index fails to capture the long-range dependencies required for the query, leading to incomplete answer.

E.3 Case Study: Entity Name vs. Entity Description as Keys

To further investigate the impact of different indexing strategies on retrieval performance, we conduct a qualitative analysis using a temporal reasoning query. As illustrated in Figure E.5, the user asks about a specific past action: "I mentioned cooking something for my friend a couple of days ago. What was it?"

Limitations of Entity Name as Key: When using the entity name as the primary retrieval key, the system tends to retrieve broad, high-level concepts. For instance, entities like "RECIPE IDEAS" or "MEAL PREP IDEAS" are successfully identified, but their associated content remains too generic—focusing on the user's general goals rather than specific past events. Furthermore, temporal entities like "COUPLE OF DAYS" are retrieved as abstract durations without being grounded to the actual context of "baking." Consequently, the model fails to locate the specific "chocolate cake" event, leading to a response that only mentions general meal preferences.

Advantages of Entity Description as Key: In contrast, using the entity description as the key allows the retriever to leverage the semantic richness

Case Study: Graph fail case(ID: b86304ba)

User Query: *"How much is the painting of a sunset worth in terms of the amount I paid for it?"*
Answer: The painting is worth triple what I paid for it.
Ground Truth Session: "user":I'm thinking of taking an art history course to learn more about the context and significance of different art movements. Can you recommend any good online resources or textbooks for beginners? "assistant":... "user":That's really helpful. I was thinking about my flea market find, and I realized that it's actually worth triple what I paid for it, which is amazing! ...

Graph Index	Flat Index
Retrieved Entities (total 20 entities): • **Entity 3:** "triple the price": The flea market find is worth triple what the user paid for it.	**Retrieved noes (total 20 memory notes):** • **Note 1:** Keywords:Organizing rare items;... Facts: The user is trying to organize... ;The user has been lucky with their flea market find and thinks it's **worth triple what they paid for it**. Summary: The user is interested in taking an art history...
• **Entity 8, 11:** "flea market": Location where the user acquired their art piece."flea market find": A piece of art acquired by the user from a flea market. • **Entity 4, 13, 20:** [Price Distractors]: "$250" for a stand mixer;"$20"for a vinyl record;"$5.25" for a bundle.	• **Note 2:** Keywords: Organizing rare items;... Facts: The user is trying to organize and research...Summary: A user is seeking help from an AI assistant to ... • **Note 3:** Keywords: Here are the keyphrases from our conversation:...Facts: The user has a vintage watch collection;...Summary: The user is a collector with a diverse collection of items...
• **Other 14 entities:** [Irrelevant Entities]:materials used, artist reputation, living room, textbooks...	• **Other 17 Notes:** [Irrelevant]:note 4:Seoul trip planning note5:Stand mixer ...[others ommited]
Model Response (Failure): *"To answer the question...3: TRIPLE THE PRICE: The flea market find is worth triple what the user paid for it....However, there is no specific information about the painting of a sunset....To clarify, I would like to ask:..."*	**Model Response (Success):** *"To answer the question... The user has been lucky with their flea market find and thinks it's worth triple what they paid for it.... Answer: It is worth triple what you paid."*

Figure E.2: **A representative case study comparing retrieval strategies in Value=Key setting.** While the graph-based index retrieves 20 atomic entities, their fragmented nature strips away essential context, leading the model to fail in entity alignment (e.g., linking the "flea market find" to the "painting"). In contrast, the flat index succeeds by providing 20 memory notes that integrate keywords, facts, and summaries. This structured representation preserves sufficient semantic cohesion, allowing the model to focus on the semantic core and reason correctly despite the presence of irrelevant distractors.

of summarized events. In this case, the system successfully retrieves the entity "LAST WEEKEND" because its description explicitly contains the specific detail: "A time period when the user made a chocolate cake for a friend's birthday party." By indexing the description, the semantic gap between the user's query ("cooking something for a friend") and the stored memory is bridged more effectively.

Conclusion: This case study demonstrates that for complex long-term memory tasks—especially those involving temporal reasoning and specific event recall—entity descriptions serve as more informative keys than entity names. Descriptions capture the unique "who, what, and when" of an interaction, whereas names often fall back on redundant or overly-categorized labels that lack the granularity required for precise retrieval.

F Prompts

Figure F.1 shows the prompt we use for graph construction. We use the official prompts from LongMemEval to extract memory (summary, factual statements and keywords). To generate the answers and judge the results, we use the official prompts from LongMemEval or HaluMem in the corresponding experiments. We use the default answering setting (CON + JSON) in LongMemEval experiments.

Case Study: Graph Success case(ID: 7a87bd0c)

User Query: *"How long have I been sticking to my daily tidying routine?"*
Answer: 4 weeks.
Ground Truth Session:
1. "user":I need help organizing my garage this weekend. Can you give me some tips on how to sort through all the boxes and storage bins? Oh, and by the way, I've been feeling really proud of myself for sticking to my daily tidying routine - it's already been **3 weeks**! "assistant":...
2."user":I'm planning to clean out the garage this weekend, but I need some advice on how to organize all the storage bins and boxes. Do you have any tips? By the way, I've been sticking to my daily tidying routine for **4 weeks now**, and it's amazing how much of a difference it's made in my apartment."assistant"...

Graph Index	Flat Index
Retrieved Sessions: • **Session 1 (Target):** *Date: 2023/06/11.* ...proud of myself for sticking to my daily tidying routine - it's already been **3 weeks**! • **Session 2-4:** [Irrelevant: Discussions on chicken stir-fry recipes, Thai green curry, and oven temperature tips.] • **Session 5 (Target):** *Date: 2023/09/30.* ...I've been sticking to my daily tidying routine for **4 weeks** now, and it's amazing...	**Retrieved Sessions:** • **Session 1 (Target):** *Date: 2023/06/11.* ...sticking to my daily tidying routine - it's already been **3 weeks**! • **Session 2-4:** [Irrelevant: Detailed recipes for stir-fry and curry; kitchen design inspiration.] • **Session 5 (Target):** *Date: 2023/09/30.* ...sticking to my daily tidying routine for **4 weeks** now, and it's amazing how much...
Model Response (Success): *"To answer the question... it is likely that the routine started around **4 weeks** before the current date, which is September 30, 2023."*	**Model Response (Failure):** *"To answer the question...Based on the information extracted, I can see that the user has been sticking to their daily tidying routine for **at least 3 weeks, and possibly 4 weeks**, depending on the session...it is clear that the user has been sticking to their daily tidying routine for at least 3 weeks."*

Figure E.3: **Case where the Graph Index succeeds in knowledge update task while the Flat Index fails.** In this instance, both strategies successfully retrieve the necessary sessions. While the narrative coherence of the flat index was advantageous in Figure E.1, this case demonstrates that such an advantage is not absolute.

Case Study: Graph Success case (ID: 5025383b)

User Query: *"What are the two hobbies that led me to join online communities?"*
Answer: Photography and Cooking.
Ground Truth Session:
1. "user":I'm looking for some advice on camera lenses.... for street **photography**. By the way, I've been really enjoying editing my photos in Lightroom - the online communities I've joined have been super helpful in learning new techniques and getting feedback on my work....
2. "user":I'm looking for some recipe inspiration...I've already joined a few online communities related to **cooking**, which led me to engage ...

GraphIndex	**FlatIndex**
Retrieved Sessions: • **Session 1 (Target):** *"I'm looking for some advice on camera lenses...."* • Session 2: I need help with creating a content calendar for my social media posts... • Session 3: What is the current state of the real estate market in New York City... • **Session 4 (Target):** *"I'm looking for some recipe inspiration..."* • Session 5: I'm looking for some advice on finding rare collectibles online...	**Retrieved Sessions:** • **Session 1:** *"I'm looking for some book recommendations..."* • Session 2: What are some effective ways to build social connections and overcome loneliness?... • Session 3: How does Doug stay up-to-date with his hobbies or interests... • **Session 4 (Target**): I'm looking for some recipe inspiration... • Session 5: I'm looking for some advice on finding rare collectibles online...
Model Response (Success): *"To answer the question...I can see that the user mentioned two hobbies that led them to join online communities: Photography (Session 1) Cooking (Session 4)...Therefore, the answer to the question is:...photography and cooking."*	**Model Response (Failure):** *"To answer the question...From the chat history, I can see that the user mentioned the following hobbies: Reading books (Session 1, Session 4) Cooking (Session 4, Session 5)...Therefore, the two hobbies... are:1. **Reading books**2. **Cooking**"*

Figure E.4: **Case where the Graph Index succeeds in multi session task while the Flat Index fails.** In this instance, the query requires aggregating evidence across disjoint and non-contiguous sessions. The **Graph Index** accurately recalls both the photography-related (Session 1) and cooking-related (Session 4) contexts. Conversely, the **Flat Index** fails to retrieve the initial photography session, leading the model to rely on irrelevant retrieved noise and subsequently hallucinate "reading books" as a hobby. This comparison highlights the robustness of graph-based indexing in maintaining high recall for long-range, cross-session dependencies.

Case Study: Precision in Temporal and Event Retrieval (ID: 9a707b82)

User Query: *"I mentioned cooking something for my friend a couple of days ago. What was it?"*
Answer: A chocolate cake.
Ground Truth Session:
Date: 2022/03/15. ...I'm excited to try making croissants again... By the way, I just baked a **chocolate cake** for my **friend's birthday party last weekend** that turned out amazing. It was a new recipe I found online...

Entity Name as Key	Entity Description as Key
Retrieved Entities & Context: • **Entity:** "COUPLE OF DAYS" (Duration) *Content:* A short period of time, approximately 2-3 days. • **Entity:** "RECIPE IDEAS" (Goal) *Content:* The user seeks innovative dessert and savory dish suggestions. • **Entity:** "MEAL PREP IDEAS" (Behavior) *Content:* User is seeking meal preparation ideas that can be reheated.	**Retrieved Entities & Context:** • **Entity:** "LAST WEEKEND" (Time) *Content:* **A time period when the user made a chocolate cake for a friend's birthday party.** • **Entity:** "DINNER PARTY" (Event) *Content:* A social gathering hosted by the user to showcase their culinary skills. • **Entity:** "ROASTED VEGETABLES" (Object) *Content:* A meal prep item made by the user last weekend.
Model Response (Failure): Based on your previous mentions, you were looking for **meal prep ideas** and **recipe ideas** for a dinner party, but there is no specific mention of what you cooked for a friend a few days ago.	**Model Response (Success):** You mentioned that **last weekend** you baked a **chocolate cake** for your friend's birthday party. You noted it was a new recipe using espresso powder.

Figure E.5: **Comparison of retrieval performance between Entity Name and Entity Description as keys.** The description-based key successfully links the temporal entity "Last Weekend" to the specific event of "baking a chocolate cake," whereas the name-based key retrieves generic categories.

Prompt for Entity Relation Extraction

Goal: Given a multi-turn conversation consisting only of the user's messages (each turn separated by "\n"), extract structured information that reflects the user's activities, possessions, goals, behaviors and reactions. Identify all relevant entities and their relationships to build a knowledge graph representing the user's context and life events.

Steps:

1. Treat the entire conversation as one continuous narrative reflecting the user's life. Integrate information across all turns to infer complete and coherent entities and relationships.
2. Identify all entities mentioned or implied by the user. For each entity, extract:
   - `entity_name`: Name of the entity, capitalized.
   - `entity_type`: One of the following types: [User, Person, Object, Resource, Event, Goal/Intention, Time, Statistic, Duration, Place, Organization, Interest/Skill, Sentiment, Health, Behavior, Other]
   - `entity_description`: A comprehensive description summarizing how this entity relates to the user and any attributes mentioned (e.g., purpose, frequency, purchase time, emotional tone).

   Format each entity as:

   ```
   ("entity"<|><entity_name><|><entity_type><|><entity_description>)
   ```

3. **Time Normalization and Extraction:** Whenever a specific or relative date is mentioned in the conversation, standardize it as a separate entity of type `"time"`. Follow these rules:
   - Use the provided conversation time `{dialogue_time}` as reference.
   - If an explicit date is mentioned (e.g., "March 2nd"), convert it to `YYYY/MM/DD` format.
   - If a relative time (e.g., "yesterday", "last week") appears, infer its absolute date relative to `{dialogue_time}`.
   - Do **not** create separate entities for recurring or habitual times (e.g., "every morning", "three times a week"); include such patterns only in related entity/relationship descriptions.
   - Each time entity should describe **what happened at/before/after that time**.
4. **Quantitative & Frequency Extraction:** Explicitly extract any quantity, count, frequency, or duration mentioned in the conversation that describes the user's actions, achievements, or possessions. Include these as separate `"Statistic"` or `"Duration"` entities. Examples:

   ```
   ("entity"<|>"Three Goals"<|>"Statistic"<|>"The user has scored 3 goals...")
   ("entity"<|>"Three Times A Week"<|>"Statistic"<|>"The user performs an activity...")
   ("entity"<|>"Five Weeks"<|>"Duration"<|>"The activity lasted for 5 weeks.")
   ```

5. From the identified entities, detect all pairs of (source_entity, target_entity) that have a meaningful or causal relationship in the context of the user's life. For each relationship, extract:
   - `source_entity`: name of the source entity
   - `target_entity`: name of the target entity
   - `relationship_description`: a natural-language description explaining the relationship or connection between the source entity and the target entity.
   - `relationship_strength`: a numeric score (1–10) estimating how strong or explicit this connection is.

   Format each relationship as:

   ```
   ("relationship"<|><source_entity><|><target_entity>
    <|><relationship_description><|><relationship_strength>)
   ```

6. Return output in English as a single list of all identified entities and relationships. Use **##** as the list delimiter.
7. When finished, output `<|COMPLETE|>`.

##############################

Real Data:

######################

```
Conversation time: {dialogue_time}
Text: {input_text}
```

######################

Output:

Figure F.1: The specific prompt used for entity and relation extraction