Implicit Graph, Explicit Retrieval: Towards Efficient and Interpretable Long-horizon Memory for Large Language Models

Anonymous ACL submission

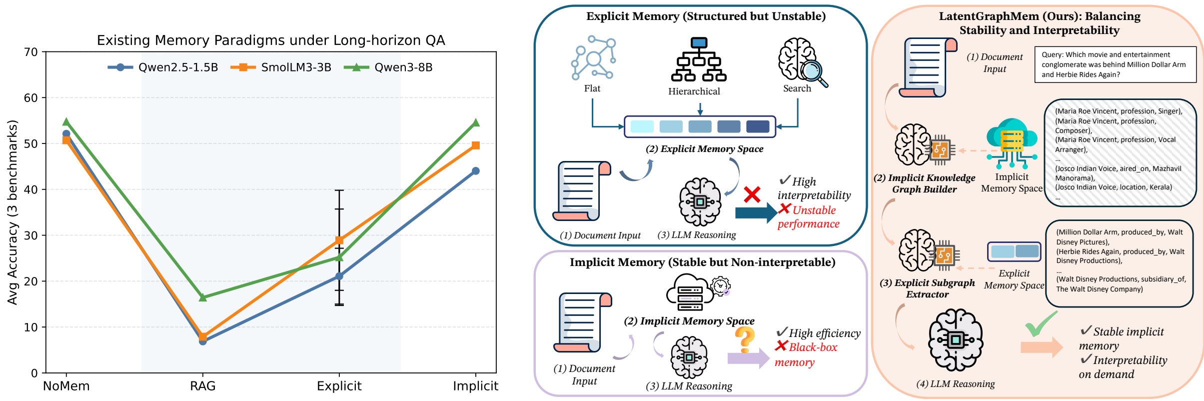
Abstract

Long-horizon applications increasingly require large language models (LLMs) to answer queries when relevant evidence is sparse and dispersed across very long contexts. Existing memory systems largely follow two paradigms: explicit structured memories offer interpretability but often become brittle under long-context overload, while latent memory mechanisms are efficient and stable yet difficult to inspect. We propose LATENTGRAPHMEM, a memory framework that combines implicit graph memory with explicit subgraph retrieval. LATENTGRAPHMEM stores a graph-structured memory in latent space for stability and efficiency, and exposes a task-specific subgraph retrieval interface that returns a compact symbolic subgraph under a fixed budget for downstream reasoning and human inspection. During training, an explicit graph view is materialized to interface with a frozen reasoner for question-answering supervision. At inference time, retrieval is performed in latent space and only the retrieved subgraph is externalized. Experiments on long-horizon benchmarks across multiple model scales show that LATENTGRAPHMEM consistently outperforms representative explicit-graph and latent-memory baselines, while enabling parameter-efficient adaptation and flexible scaling to larger reasoners without introducing large symbolic artifacts.

1 Introduction

Memory mechanisms have become a foundational component of modern large language model (LLM) systems (Hu et al., 2025). In applications such as long-chain reasoning and multi-agent settings, carefully designed memory architectures enable LLMs to accumulate, organize, and reuse complex information beyond the constraints of a single prompt or fixed context window. Existing approaches can be broadly categorized into two dominant paradigms. The first paradigm consists of explicitly structured memory designs (Noohi et al., 2025), with particular emphasis on graph-structured representations (Huang et al., 2025; Chhikara et al., 2025; Ong et al., 2025). The second paradigm encompasses implicit memory mechanisms, which store and manipulate information directly within continuous latent representations (Zhang et al., 2025; Wang et al., 2025b,a). Graph-based explicit memory mechanisms offer notable advantages in interpretability, as they facilitate inspection, debugging, and task-relevant memory graph retrieval (Rasmussen et al., 2025; Anokhin et al., 2025; Gutierrez et al., 2024). In contrast, latent memory mechanisms demonstrate high efficiency, requiring substantially lower storage and computational overhead during memory usage.

In this work, we focus on the important long-horizon memory scenarios(Bai et al., 2025; Moon and Lim, 2025), where LLMs must operate over multi-thousand-token contexts, in which task-relevant evidence is sparsely scattered across distant segments of the memory. To compare the two memory paradigms, we conduct preliminary experiments on three representative long-horizon memory benchmarks and evaluate five representative memory mechanisms, including MemGen(Zhang et al., 2025), A-Mem(Xu et al., 2025), PREMem(Kim et al., 2025), THEANINE(Ong et al., 2025), and Mem0(Chhikara et al., 2025). As illustrated in Figure 1a, explicit structured memory approaches frequently underperform even a vanilla baseline memory system across multiple tasks. This instability is primarily caused by long-context overload: (i) structure induction becomes unreliable when the input is long and noisy, (ii) retrieval becomes increasingly sensitive to query variations as the symbolic space grows. In contrast, implicit memory mechanisms yield more consistent improvements. Nevertheless, their principal limitation lies in low interpretability, as it remains challenging to inspect the stored representations or to explain why spe-



(a) (b)

Figure 1: (a) Performance of existing memory paradigms under long-horizon QA. **NoMem**: reasoner-only; **RAG**: retrieval-augmented generation; **Explicit**: explicit structured memory systems (A-Mem, PREMem, THEANINE, Mem0); **Implicit**: latent memory (MemGen). (b) Explicit and implicit memory paradigms for long-horizon reasoning. Existing explicit structured memory systems are interpretable but brittle under long and dispersed evidence, while implicit memory is more stable but opaque. LATENTGRAPHMEM combines implicit graph memory with explicit subgraph exposure.

cific pieces of evidence are retrieved during task execution.

The above analysis highlights the limitations of relying exclusively on either fully explicit structured memories or purely latent memory paradigms in long-horizon settings. Motivated by these observations, we revisit a practical assumption in memory design: while an explicit symbolic view is often needed to support interpretable evidence access for downstream reasoning, operating on explicit structured alone can be brittle and inefficient under long and noisy contexts. We propose LATENTGRAPHMEM, a simple yet effective memory framework that combines the complementary strengths of both paradigms: it learns a graph memory with latent representations for efficient retrieval and control, while enabling task-specific *explicit* subgraph exposure to support interpretability. Figure 1b illustrates the conceptual advantages of LATENTGRAPHMEM over previous approaches. LATENTGRAPHMEM consists of two core components. First, a **graph memory builder** is trained via parameter-efficient adaptation to construct a bounded symbolic graph view from long documents in a streaming manner, and to produce latent embeddings for the resulting graph edges. Second, a **subgraph memory retriever** is adapted to select a compact evidence subgraph from these latent edge embeddings under a fixed retrieval budget, which is then externalized and fed to a frozen reasoner for answering. Notably, LATENTGRAPHMEM performs retrieval in latent space after the builder is trained, and at inference time it exposes only a compact, interpretable evidence subgraph to the reasoner rather than the full graph.

Extensive experiments on three long-horizon QA benchmarks demonstrate that LATENTGRAPHMEM consistently outperforms state-of-the-art graph-based and latent memory baselines across frozen-reasoner scales ranging from *1.5B* to *8B*. In particular, LATENTGRAPHMEM achieves the best average accuracy at every scale, reaching 56.08%, 58.64%, and 63.34% under Qwen2.5-1.5B, SmolLM3-3B, and Qwen3-8B, respectively, surpassing the strongest competing memory system. These results confirm that encoding graph-structured memory implicitly while exposing compact explicit subgraphs enables robust and scalable long-horizon reasoning. Overall, this paper makes three primary contributions: (1) a systematic empirical study of explicit and implicit memory paradigms in long-horizon settings; (2) LATENTGRAPHMEM, a simple framework that unifies latent graph storage with explicit subgraph retrieval; and (3) consistent performance gains across model scales while maintaining interpretability through explicit retrieved evidence.

2 Related Work

2.1 Explicit Structured Memory and Graph-Based Retrieval

A dominant line of agent memory research builds an *explicit* external store (often text, key–value entries, or databases) and retrieves from it at test time. Representative systems include flat or database-style memories such as MemGPT (Packer et al., 2023), as well as *structured* memories that impose relational and temporal organization to support controlled retrieval and updates. Recent graph-oriented agent memories maintain entity- or chunk-level nodes with explicit links, including production-focused long-term memory systems with an optional graph variant (Mem0 / Mem0g) (Chhikara et al., 2025) and dynamic indexing/linking designs (A-Mem) (Xu et al., 2025) that build an explicit memory network for retrieval-time access (Xu et al., 2025). Beyond general graph stores, several works emphasize *graph-based retrieval* with higher-level structure: GraphRAG (Edge et al., 2024) constructs a multi-level graph index via community detection and recursive summarization, and Zep (Rasmussen et al., 2025) models agent memory as a temporal knowledge graph with community partitioning. For long-term dialogue, THEANINE (Ong et al., 2025) organizes stored experiences along explicit temporal links and retrieves coherent timelines rather than isolated top-k items. PREMem further strengthens explicit memory construction by performing pre-storage reasoning to form higher-quality structured memories for downstream retrieval and generation (Kim et al., 2025). These explicit-structure approaches are interpretable, but their end-to-end performance is highly sensitive to errors in extraction, indexing, and retrieval, and they require re-serializing structured artifacts into the token space for reasoning.

2.2 Latent and Implicit Memory in the Representation Space

In parallel, a second family of approaches stores memory implicitly in the model's internal representations rather than human-readable tokens. A recent survey terms this latent memory, and organizes methods by how the latent state is formed and introduced, including Generate, Reuse, and Transform mechanisms (Hu et al., 2025). Representative Generate methods compress long-horizons into learnable soft representations, such as AutoCompressor (Chevalier et al., 2023) and MemoRAG (Qian et al., 2025), while MemoryLLM maintains persistent latent tokens for factual memory (Wang et al., 2024). More architecturally integrated designs extend latent memory across layers (e.g., M+) or introduce structured latent slots (e.g., LM2) (Wang et al., 2025a; Kang et al., 2025). A different branch internalizes memory production into parameter dynamics, e.g., Titans (Behrouz et al., 2025). Closer to our setting, MemGen dynamically generates latent memory during decoding via lightweight adaptation modules (Zhang et al., 2025). Compared with explicit stores, latent memory can preserve fine-grained information and avoid excessive prompt expansion, but it remains largely opaque and difficult to interpret.

3 Method

We propose LATENTGRAPHMEM, a three-stage framework for long-context question answering that leverages a learned graph memory to support reasoning, corresponding to graph construction, subgraph retrieval, and joint refinement. Given a long document x and a question q, our goal is to construct and utilize structured evidence that enables a pretrained language model to generate accurate answers. LATENTGRAPHMEM consists of two trainable agents and a frozen reasoner. A graph builder constructs a global graph representation from the document, while a subgraph retriever selects question-relevant parts of this graph under a fixed budget. The selected evidence is then provided to an LLM for answer generation.

Formally, the frozen reasoner F defines a conditional distribution $p_F(\cdot \mid \cdot)$. We use $\oplus$ to denote string concatenation (i.e. prompt composition). The builder B_ϕ outputs an edge set $\mathcal{E}$ (triples) from x, and the retriever R_ψ selects a subgraph $S_q \subseteq \mathcal{E}$ conditioned on q. Each training instance is (x, q, a), where $a = (a_1, \ldots, a_{|a|})$ denotes the gold answer token sequence.

3.1 Stage I: Remote-Supervised Full-Graph Construction

Streaming graph construction for long documents. Context in long-horizon scenario x is usually long, so we split it into overlapping chunks $\{x^{(c)}\}_{c=1}^{C}$ with maximum length L tokens (overlap O). The builder processes chunks sequentially and maintains an evolving explicit graph state:

$$\begin{aligned} \mathcal{G}^{(0)} &= \emptyset, \\ Y^{(c)} &= \mathrm{Extract}_\phi(x^{(c)}), \\ \tilde{\mathcal{G}}^{(c)} &= \mathrm{Merge}\Big(\mathcal{G}^{(c-1)}, Y^{(c)}\Big), \\ \mathcal{G}^{(c)} &= \mathrm{Cap}\Big(\mathrm{Filter}\Big(\tilde{\mathcal{G}}^{(c)}\Big)\Big). \end{aligned} \tag{1}$$

$\mathrm{Extract}_\phi$ emits a small set of candidate triples from chunk $x^{(c)}$. We denote the edge set of $\mathcal{G}^{(c)}$

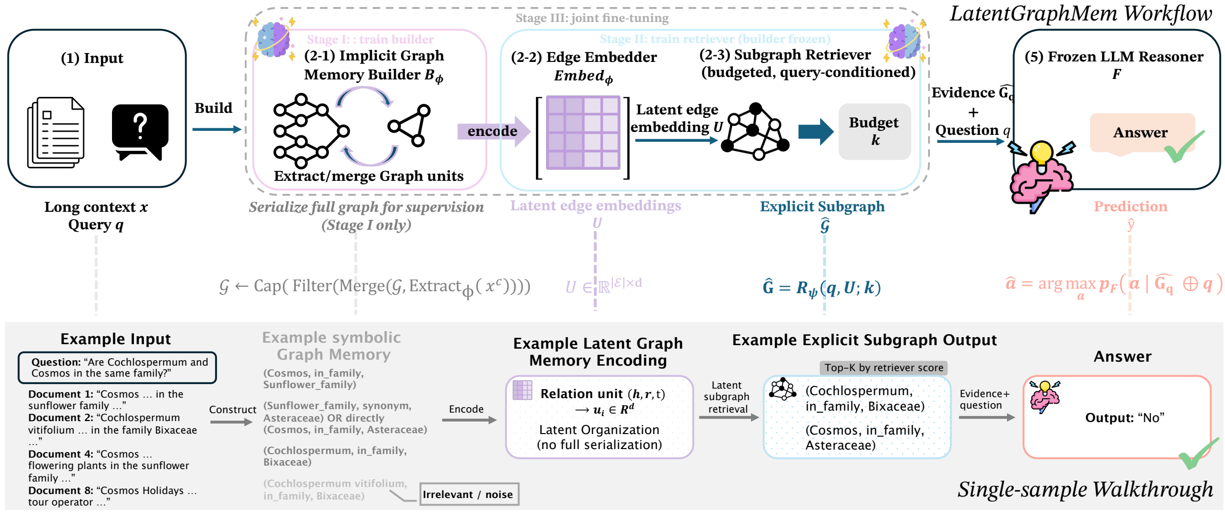


Figure 2: **Overview of LatentGraphMem.** Given a long-horizon x and query q, LatentGraphMem builds an implicit graph-structured memory from x, extracts a compact explicit subgraph as evidence conditioned on q, and answers with a frozen LLM reasoner using only the question and the extracted subgraph. The context is used only for memory construction, and the lower panel shows a single-sample walkthrough.

as $\mathcal{E}^{(c)}$. After processing all chunks, we define the final graph as $\mathcal{G} = \mathcal{G}^{(C)}$ and its edge set as $\mathcal{E} = \mathcal{E}^{(C)}$. Merge canonicalizes and deduplicates entities/relations across chunks. Filter enforces quality constraints (e.g., schema validity, field-length limits, duplicate removal, and relation-type constraints). Cap enforces a hard capacity limit to prevent graph explosion. Concretely, Cap enforces $|\mathcal{E}^{(c)}| \leq M$ at every step c. For each chunk, the builder emits a constrained triple set:

$$Y^{(c)} = \{(h_j, r_j, t_j)\}_{j=1}^{|Y^{(c)}|}, \ |Y^{(c)}| \leq m_{\text{chunk}}, \quad (2)$$

where m_{chunk} is a fixed per-chunk extraction cap. We additionally apply hard constraints: (i) maximum token length for each field h, r, t, and (ii) a global edge cap M such that the final explicit graph $\mathcal{G} = \mathcal{G}^{(C)} = (\mathcal{V}, \mathcal{E})$ satisfies $|\mathcal{E}| \leq M$. Here $\mathcal{V}$ denotes the set of unique entities appearing in $\mathcal{E}$. These controls ensure stable graph size and maintain a consistent evidence budget.

Reasoner interface (explicit full graph) and QA-only SFT loss. Stage I trains the builder using only the downstream answer objective. We serialize the explicit full graph into a deterministic evidence string: $\hat{G} = \mathrm{Serialize}(\mathcal{E})$, where Serialize formats edges using a fixed template (e.g., `Relevant Knowledge: [h|r|t] ...`). Serializing the full graph $\hat{G}$ is used only in Stage I to interface with the reasoner for supervision; at inference time, we serialize only the retrieved subgraph S_q. The frozen reasoner F defines an autoregressive conditional distribution $p_F(\cdot)$ over answer tokens conditioned on the serialized graph evidence and the question. We optimize the builder parameters ϕ by minimizing the cross-entropy loss:

$$\mathcal{L}_{\mathrm{I}}(\phi) = -\sum_{t=1}^{|a|} \log p_F\Big(a_t \mid a_{<t}, \hat{G} \oplus q\Big), \quad (3)$$

where $a_{<t} = (a_1, \ldots, a_{t-1})$ denotes the prefix of the gold answer under teacher forcing.

3.2 Stage II: Remote-Supervised Latent Subgraph Retrieval

Implicit graph representation after Stage I. Although Stage I materializes an explicit graph to interface with the reasoner, after training we use an implicit representation for downstream control. Specifically, each retained edge $e_i = (h_i, r_i, t_i) \in \mathcal{E}$ is mapped to a latent embedding:

$$u_i = \mathrm{Embed}_\phi(h_i, r_i, t_i) \in \mathbb{R}^d, \ i = 1, \ldots, |\mathcal{E}|. \quad (4)$$

We store the edge embeddings as a matrix $U \in \mathbb{R}^{|\mathcal{E}| \times d}$, where the i-th row corresponds to edge e_i and equals u_i. Stage II and Stage III perform selection in this latent space.

Training regime and inputs/outputs. Stage II starts from the builder trained in Stage I and fixes B_ϕ. Given (q, U), the subgraph retriever R_ψ selects a compact subset of edges under a budget k, operating only on the embeddings U. The output is an index set $\mathcal{I}_q$ and the corresponding edge subset $S_q = \{e_i \mid i \in \mathcal{I}_q\}$.

Latent scoring and budgeted selection. We encode the question and compute relevance scores for each edge embedding:

$$v = R_{\psi}^{\text{enc}}(q) \in \mathbb{R}^d,\ s_i = v^\top W u_i,\ W \in \mathbb{R}^{d \times d}. \tag{5}$$

The bilinear weight W is a learnable parameter of the retriever and is included in ψ. We then select at most k edges:

$$\mathcal{I}_q = \text{TopK}(\{s_i\}, k),\ S_q = \{e_i \mid i \in \mathcal{I}_q\}. \tag{6}$$

This provides explicit budget control and yields a compact subgraph.

Differentiable training path (SFT, no RL). Since TopK is not differentiable, we use a straight-through estimator: the forward pass uses the hard top-k selection, while the backward pass routes gradients through a softmax relaxation with temperature τ:

$$z = \text{TopKMask}(\{s_i\}, k),\ \alpha_i = \frac{\exp(s_i/\tau)}{\sum_j \exp(s_j/\tau)}. \tag{7}$$

We form S_q from z in the forward pass and use α for gradient flow in the backward pass. This keeps training within standard backpropagation and does not require reinforcement learning.

Reasoner interface (explicit retrieved subgraph) and QA-only SFT loss. We externalize only the retrieved subgraph for the reasoner:

$$\hat{G}_q = \text{Serialize}(S_q), \tag{8}$$

and train the retriever using the same QA-only cross-entropy loss:

$$\mathcal{L}_{\text{II}}(\psi) = -\sum_{t=1}^{|a|} \log p_F\Big(a_t \mid a_{<t},\ \hat{G}_q \oplus q\Big). \tag{9}$$

3.3 Stage III: Joint Fine-Tuning

Training regime. Stage III starts from the builder obtained in Stage I and the retriever obtained in Stage II. We jointly fine-tune (ϕ, ψ) to maximize their coordination: the builder should produce edges/embeddings that are easy to retrieve and useful when externalized, and the retriever should best exploit the builder's latent space. The reasoner F remains frozen.

End-to-end dataflow and objective. The full computation is:

$$x \xrightarrow{B_\phi} (\mathcal{E}, U) \xrightarrow{R_\psi(q)} S_q \xrightarrow{\text{Serialize}} \hat{G}_q \xrightarrow{F} a, \tag{10}$$

and we optimize the QA-only SFT objective:

$$\mathcal{L}_{\text{III}}(\phi, \psi) = -\sum_{t=1}^{|a|} \log p_F\Big(a_t \mid a_{<t},\ \hat{G}_q \oplus q\Big). \tag{11}$$

In practice, we use stable alternating updates: builder-only steps minimizing $\mathcal{L}_{\text{I}}$ to preserve Stage I construction quality, and joint steps minimizing $\mathcal{L}_{\text{III}}$ to improve coupling. Concretely, builder-only steps use the full-graph interface $\hat{G} = \text{Serialize}(\mathcal{E})$ as an auxiliary objective to preserve extraction quality, whereas joint steps use the retrieved subgraph interface $\hat{G}_q = \text{Serialize}(S_q)$ to improve builder–retriever coupling.

Inference. At test time, given (x, q), the builder streams over chunks and maintains an internal graph state, producing an edge set $\mathcal{E}$ and an embedding matrix $U \in \mathbb{R}^{|\mathcal{E}| \times d}$ without serializing the full graph into the prompt. Only the retrieved subgraph S_q is serialized and passed to the frozen reasoner. The retriever selects a budgeted subgraph $S_q \subseteq \mathcal{E}$ in latent space, which is then serialized as evidence for the frozen reasoner:

$$\hat{a} = \arg\max_a p_F(a \mid \text{Serialize}(S_q) \oplus q). \tag{12}$$

4 Experiments

4.1 Datasets

We construct a long-horizon QA training corpus of 20,800 instances from three real-world sources (TriviaQA (Joshi et al., 2017), QASPER (Dasigi et al., 2021), and QuALITY (Pang et al., 2022)) and evaluate on a fixed 2,600-instance mixture from HotpotQA (Yang et al., 2018), NarrativeQA (Kociský et al., 2018), and WikiHop (Welbl et al., 2018). Table 3 reports dataset statistics computed on our processed splits, including character counts and tokenizer-specific token counts under each base model for consistent measurement across configurations. Additional statistics and dataset characteristics are provided in Appendix A.1. **Training datasets.** TriviaQA provides open-domain questions with long, heterogeneous evidence collections, QASPER is grounded in full scientific papers with structured long contexts, and QuALITY

Table 1: Main results on the merged QA test set (Acc/ROUGE-L, higher is better; shown in %). Avg is the unweighted mean over the three datasets. We highlight best/second/worst for each column (bold/underline/gray).

Method	HotpotQA		NarrativeQA		WikiHop		Avg	
	Acc	R-L	Acc	R-L	Acc	R-L	Acc	R-L
Backbone: Qwen2.5-1.5B-Instruct								
Reasoner-only	80.20	52.20	41.50	28.13	34.63	23.57	52.11	34.63
MemGen	57.10	57.59	42.63	**54.20**	32.37	**32.54**	44.03	**48.11**
A-Mem	46.60	56.73	15.62	49.85	12.12	20.53	24.78	42.37
PREMem	49.70	48.78	10.37	11.64	10.13	10.78	23.40	23.73
THEANINE	19.40	8.19	2.25	3.84	10.25	6.64	10.63	6.22
Mem0	63.80	44.08	1.12	5.23	11.63	10.15	25.52	19.82
RAG	14.60	29.05	4.50	26.47	1.50	8.05	6.87	21.19
LatentGraphMem (Ours)	**86.60**	**58.54**	**45.50**	32.25	**36.13**	25.03	**56.08**	38.61
Backbone: SmolLM3-3B								
Reasoner-only	77.10	51.46	40.50	37.43	34.50	24.61	50.70	37.83
MemGen	58.00	53.64	49.38	47.37	41.35	**42.59**	49.58	47.87
A-Mem	51.30	61.18	19.88	**55.99**	17.13	28.45	29.44	**48.54**
PREMem	87.70	60.00	11.22	9.11	38.50	25.77	45.81	31.63
THEANINE	42.79	19.93	10.87	7.84	19.75	13.36	24.47	13.71
Mem0	44.90	29.30	0.25	3.04	2.50	2.26	15.88	11.53
RAG	15.30	52.22	6.12	40.12	2.25	16.52	7.89	36.29
LatentGraphMem (Ours)	**88.90**	**61.39**	**47.00**	35.38	**40.03**	35.71	**58.64**	44.16
Backbone: Qwen3-8B								
Reasoner-only	72.60	62.48	**51.00**	63.81	40.62	31.55	54.74	52.61
MemGen	72.30	71.03	50.88	**63.94**	40.50	40.30	54.56	**58.42**
A-Mem	31.40	55.10	15.88	46.04	9.38	17.89	18.89	39.68
PREMem	81.46	56.53	2.57	3.00	45.87	31.55	43.30	30.36
THEANINE	39.94	16.02	6.63	5.13	16.15	10.48	20.91	10.54
Mem0	48.60	29.43	0.50	3.12	4.13	3.46	17.74	12.00
RAG	29.30	63.88	14.87	47.29	5.12	20.04	16.43	43.74
LatentGraphMem (Ours)	**90.20**	**71.21**	**51.00**	37.60	**48.81**	**40.37**	**63.34**	49.73

offers multiple-choice supervision over long articles, together covering diverse long-horizon reasoning patterns. **Evaluation datasets.** We evaluate only on HotpotQA, NarrativeQA, and WikiHop, where HotpotQA and WikiHop stress multi-hop reasoning across documents and entities and NarrativeQA probes long-form narrative understanding, and none of these evaluation datasets are included in training.

4.2 Baselines

We compare LatentGraphMem with seven baselines. Reasoner-only (full context) answers directly from the full input (up to the context limit) without any external memory, while RAG retrieves top-k chunks from the input via embedding similarity and appends them as textual evidence to the same frozen reasoner. In addition, we include five representative memory-system baselines covering both explicit structure and latent memory: THEANINE (timeline-style graph memory management) (Ong et al., 2025), PREMem (pre-storage reasoning for episodic memory) (Kim et al., 2025), Mem0 / Mem0g (production-oriented long-term memory with an optional graph variant) (Chhikara et al., 2025), A-Mem (agentic memory with dynamic indexing and linking) (Xu et al., 2025), and MemGen (generative latent memory tokens) (Zhang et al., 2025).

Implementation Details. For all experiments, the graph builder and subgraph retriever are instantiated as decoder-only LLMs with parameter-

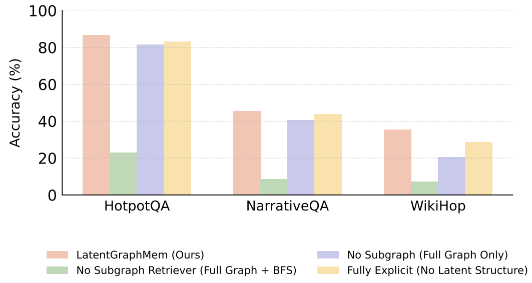


Figure 3: Memory Structure Ablation

Table 2: Inference time (seconds) versus context length (k tokens) for long-context samples ($\geq$6k).

Context (k)	Ours (s)	MemGen (s)	A-Mem (s)
6k	12.47	10.59	20.00
7k	13.42	11.15	17.48
8k	12.84	15.52	16.50
9k	13.43	10.86	44.65
10k	13.78	8.95	41.90

efficient LoRA adaptation, while the reasoner is kept frozen throughout training and inference. In Stage I, long documents are processed in overlapping chunks of 1,024 tokens with an overlap of 128 tokens. We cap extraction to 32 triples per chunk and limit the global graph capacity to $M = 150$ edges to control memory growth. In Stage II, the retriever selects a compact evidence subgraph under a fixed budget of $k = 30$ edges, which is serialized and provided to the frozen reasoner for answer generation. We report answer accuracy (Acc) and ROUGE-L on all benchmarks. The graph builder and subgraph retriever are instantiated with Qwen2.5-1.5B-Instruct, and we evaluate with frozen reasoners at 1.5B, 3B, and 8B scales by directly reusing the same trained memory modules without additional training.

4.3 Main Results

We evaluate LatentGraphMem against seven baselines on three long-horizon QA benchmarks (HotpotQA, NarrativeQA, and WikiHop) under three frozen-reasoner backbones (Qwen2.5-1.5B, SmolLM3-3B, and Qwen3-8B), with results summarized in Table 1. Across all settings, LatentGraphMem consistently achieves the strongest overall performance in terms of average accuracy.

Specifically, (1) LatentGraphMem improves average accuracy over all baselines at every model scale, raising the average accuracy to 56.08% with Qwen2.5-1.5B and further to 63.34% with Qwen3-8B, demonstrating that the proposed memory mechanism scales reliably with increasing reasoning capacity. (2) Fully explicit graph-based memory methods, including THEANINE, PREMem, and Mem0/Mem0g, exhibit severe performance degradation in long-horizon settings, particularly on NarrativeQA, where they often collapse to near-zero or single-digit accuracy, resulting in substantially lower averages than both the reasoner-only baseline and latent-memory approaches. (3) Compared to purely latent memory, represented by MemGen, LatentGraphMem achieves consistently higher accuracy on multi-hop benchmarks such as HotpotQA across all backbones, despite MemGen occasionally attaining higher ROUGE-L scores on NarrativeQA. Notably, LatentGraphMem employs a single 1.5B graph builder that generalizes directly to larger frozen reasoners, whereas MemGen requires the memory module and reasoner to be trained at comparable scales to maintain latent alignment.

4.4 Structural Ablation on Memory Representation and Retrieval

To study the role of different memory structures and retrieval strategies in LatentGraphMem, we conduct a series of structural ablations summarized in Figure 3. Specifically, we compare our full model with variants that (i) remove subgraph retrieval and reason over the full graph directly, (ii) replace the learned subgraph retriever with a heuristic BFS expansion, and (iii) rely entirely on explicit graph construction and retrieval without latent representations. The results show that directly reasoning over the full graph degrades performance, while heuristic retrieval introduces substantial noise. Fully explicit designs perform better than naive retrieval but remain inferior to our approach, highlighting the necessity of combining latent graph memory with learned, budgeted subgraph retrieval.

4.5 Inference Time under Long Contexts

We evaluate the inference-time behavior of different memory systems under increasing context lengths to assess their scalability in long-context settings. Specifically, we measure average inference time on 100 test samples per context length using 1.5B models trained for each method. As shown in Table 2, A-Mem exhibits highly unstable inference time as the context grows, with latency increasing sharply to 44.65 seconds at 9k tokens. In contrast, LATENTGRAPHMEM maintains stable inference time across all evaluated context lengths and closely matches the efficiency of MemGen. This result indicates that, despite exposing explicit subgraph evidence for reasoning, the latent graph memory and fixed retrieval budget in LATENTGRAPHMEM prevent inference cost from scaling with the full context length, enabling predictable and scalable inference under long-context workloads.

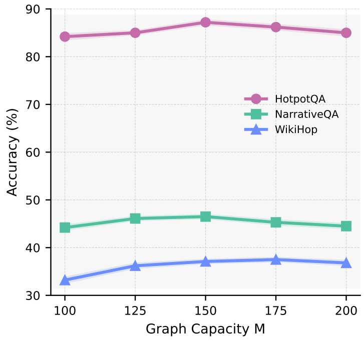


Figure 4: Effect of graph capacity M on answer accuracy across three QA datasets.

4.6 Effect of Global Graph Capacity

We analyze the impact of the global graph capacity M on model performance to understand the trade-off between memory coverage and noise. Figure 4 reports results on a 1.5B backbone as we vary M around the default setting (M=150). We observe that increasing M consistently benefits WikiHop, while the gains on HotpotQA and NarrativeQA saturate or slightly decline beyond the default value. This behavior aligns with dataset characteristics: WikiHop requires aggregating evidence across a larger set of candidate entities and relations, making broader graph coverage advantageous. In contrast, HotpotQA and NarrativeQA typically involve fewer relevant entities but more localized or narrative-driven evidence, where larger graphs introduce additional noise that can hinder retrieval and reasoning.

4.7 Case Study

We conduct a qualitative case study to examine how different memory paradigms handle sparse and dispersed evidence in long-horizon question answering. The example considers a temporal comparison between two similarly named documentaries, where the relevant release dates are distributed

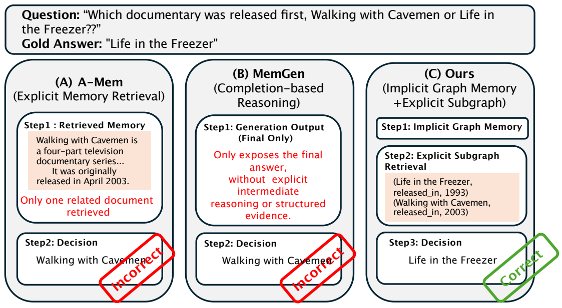


Figure 5: Case Study: Both Explicit and Implicit Construction Baselines Fail, While Our Method Succeeds

across a long and noisy context. As illustrated in Figure 5, explicit memory retrieval (A-Mem) retrieves evidence for only *Walking with Cavemen* (2003) and fails to retrieve the corresponding date for *Life in the Freezer* (1993), leading to an incorrect comparison. Completion-based latent memory (MemGen) produces a final answer without exposing intermediate evidence, making the underlying temporal reasoning unverifiable and resulting in an incorrect decision. In contrast, LATENTGRAPHMEM preserves both entities and their release years in an implicit graph memory and retrieves a compact explicit subgraph containing the relevant dates, enabling a reliable comparison and the correct answer.

5 Conclusion

We present LATENTGRAPHMEM, a memory framework for long-horizon question answering that integrates latent graph memory with explicit, budgeted subgraph retrieval. By decoupling memory construction from reasoning, our approach combines the stability and efficiency of latent representations with the interpretability and control of explicit evidence exposure. Extensive experiments across multiple long-horizon benchmarks demonstrate that LATENTGRAPHMEM achieves stronger and more consistent performance than representative explicit and latent memory baselines while maintaining stable inference time and supporting flexible scaling to larger reasoners. These results highlight the importance of structured yet implicit memory representations for reliable reasoning under long and dispersed contexts.

Limitations

While LATENTGRAPHMEM is designed for long-horizon reasoning with sparse evidence, its performance depends on the quality of the learned graph construction and retrieval modules. Although we adopt fixed budgets and capacity constraints to ensure stability and efficiency, the optimal configuration may vary across tasks with different evidence structures. In addition, our framework currently focuses on text-based inputs and question answering settings, and extending the approach to other modalities or interactive reasoning scenarios remains an interesting direction for future work.

References

Petr Anokhin, Nikita Semenov, Artyom Y. Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. 2025. Arigraph: Learning knowledge graph world models with episodic memory for LLM agents. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 12–20. ijcai.org.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 3639–3664. Association for Computational Linguistics.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. 2025. Titans: Learning to memorize at test time. *CoRR*, abs/2501.00663.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics.

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

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4599–4610. Association for Computational Linguistics.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *CoRR*, abs/2404.16130.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

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*.

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

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Jikun Kang, Wenqi Wu, Filippos Christianos, Alex J Chan, Fraser Greenlee, George Thomas, Marvin Purtorab, and Andy Toulis. 2025. Lm2: Large memory models. *arXiv preprint arXiv:2502.06049*.

Sangyeop Kim, Yohan Lee, Sanghwa Kim, Hyunjong Kim, and Sungzoon Cho. 2025. Pre-storage reasoning for episodic memory: Shifting inference burden to memory for personalized dialogue. *CoRR*, abs/2509.10852.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328.

Hyeonseok Moon and Heuiseok Lim. 2025. Needlechain: Measuring intact long-context reasoning capability of large language models. *CoRR*, abs/2507.22411.

Amir Noohi, Mostafa Derispour, and Antonio Barbalace. 2025. RMAI: rethinking memory for AI (inference): In-kernel remote shared memory as a software alternative to CXL. In *Proceedings of the 5th Workshop on Machine Learning and Systems, EuroMLSys 2025, World Trade Center, Rotterdam, The Netherlands, 30 March 2025- 3 April 2025*, pages 122–131. ACM.

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, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 8631–8661. Association for Computational Linguistics.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. 2022. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5336–5358. Association for Computational Linguistics.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. 2025. Memorag: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025-2 May 2025*, pages 2366–2377. ACM.

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*.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian J. McAuley. 2024. MEMORYLLM: towards self-updatable large language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian J. McAuley, Dan Gutfreund, Rogério Feris, and Zexue He. 2025a. M+: extending memoryllm with scalable long-term memory. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

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

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Trans. Assoc. Comput. Linguistics*, 6:287–302.

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*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. 2025. Memgen: Weaving generative latent memory for self-evolving agents. *CoRR*, abs/2509.24704.

A Appendix

A.1 Dataset Statistics and Characteristics

This appendix reports detailed statistics and characteristics of the datasets used in our experiments. Table 3 lists the number of instances and the average lengths of contexts and answers for each dataset. All statistics are computed on our processed splits: character counts are measured on the preprocessed raw text, and token counts are computed using the corresponding base-model tokenizer to ensure comparability across model configurations.

Training datasets. **TriviaQA** (Joshi et al., 2017) contains open-domain questions paired with large evidence collections, where contexts are typically long and heterogeneous after preprocessing, making it suitable for training memory mechanisms that must aggregate dispersed evidence. **QASPER** (Dasigi et al., 2021) is document-grounded QA over full scientific papers, where contexts exhibit structured regularities (e.g., sections and citations) and questions often require integrating information across distant sections. **QuALITY** (Pang et al., 2022) provides multiple-choice questions over long narrative or expository articles, where correct options frequently depend on global comprehension rather than local lexical matches, complementing extractive or short-answer supervision.

Evaluation datasets. **HotpotQA** (Yang et al., 2018) and **WikiHop** (Welbl et al., 2018) are multi-hop benchmarks requiring reasoning across multi-

Dataset	Split	#Inst.	Context avg.	Answer avg.
Training corpora				
TriviaQA (Joshi et al., 2017)	Train	14,679	75,314 chars (18,828 tok)$^{\uparrow}$	9.9 chars (2.1 tok)
QASPER (Dasigi et al., 2021)	Train	3,598	24,608 chars (6,152 tok)	6.0 chars (1.0 tok)$^{\downarrow}$
QuALITY (Pang et al., 2022)	Train	2,523	27,018 chars (6,754 tok)	49.4 chars (12.0 tok)$^{\uparrow}$
Evaluation corpora				
HotpotQA (Yang et al., 2018)	Test	1,000	5,240 chars (1,310 tok)	11.0 chars (2.7 tok)
NarrativeQA (Kociský et al., 2018)	Test	800	3,860 chars (965 tok)$^{\downarrow}$	26.0 chars (6.5 tok)
WikiHop (Welbl et al., 2018)	Test	800	7,230 chars (1,807 tok)	12.3 chars (3.1 tok)
Total	–	23,400	5,428 chars (1,356 tok)†	16.0 chars (4.0 tok)†

Table 3: Datasets used for training and evaluation. “tok” denotes token counts under the corresponding base-model tokenizer. ↑ / ↓ mark unusually long/short averages within the table (Context or Answer). †For the total row, we report the average over the test mixture (2,600 instances), consistent with our main evaluation protocol.

ple documents or entities, and **NarrativeQA** (Kociský et al., 2018) focuses on long-form narrative understanding with evidence distributed across stories. We use fixed test splits from these datasets and do not include them in training.

Test mixture. Our aggregate evaluation follows a fixed 2,600-instance mixture consisting of 1,000 HotpotQA, 800 NarrativeQA, and 800 WikiHop instances. The *Total* row in Table 3 reports length statistics averaged over this mixture, consistent with our main evaluation protocol.

A.2 Supplementary Experiment Analysis

Dataset-level trends. LatentGraphMem shows the largest accuracy gains on datasets requiring compositional reasoning over dispersed evidence. It consistently performs best on HotpotQA across all backbones, and its advantage on WikiHop becomes more pronounced as the frozen reasoner scales. On NarrativeQA, where answers are longer and more abstractive, methods tend to differ more in ROUGE-L than in exact-match accuracy; nevertheless, LatentGraphMem remains competitive in accuracy while still exposing interpretable retrieved subgraphs.

Failure modes of fully explicit graph-based memory. Across backbones, fully explicit graph-memory systems (e.g., THEANINE, PREMem, and Mem0/Mem0g) frequently collapse on NarrativeQA, producing near-zero or single-digit accuracy despite occasional competitiveness on HotpotQA. This pattern suggests that under long horizons, errors introduced by symbolic extraction and retrieval can dominate the downstream reasoning signal, so scaling the frozen reasoner alone does not reliably recover performance. In contrast, latent-based approaches are more stable under long contexts, and LatentGraphMem further improves robustness by combining latent storage with explicit subgraph exposure.