LatentGraphMem: Bridging Symbolic Memory and Latent Thinking for Large Language Models

Anonymous ACL submission

Abstract

Long-context question answering requires large language models to recover sparse evidence from noisy and dispersed contexts. Existing memories often trade interpretability for stability: explicit graph memories are inspectable but brittle, while latent memories are compact but opaque. We propose LATENTGRAPHMEM, a symbolic-latent memory framework for long-context question answering. A distilled graph-memory builder maintains bounded symbolic edges with aligned edge embeddings. A task-conditioned retriever selects latent edge rows, and their paired triples are serialized for a frozen reasoner. A row-aligned interface decouples memory construction and retrieval from the frozen LLM, allowing the same memory layer to transfer across reasoner scales. Training distills the builder from document-only teacher triples, then freezes it to learn answer-grounded retrieval over builder-induced symbolic-latent memory. Experiments on HotpotQA, NarrativeQA, and WikiHop show EM, F1, and ROUGE-1 performance across three frozen reasoner scales. Ablations and evidence analyses confirm that latent retrieval controls the evidence exposed to the reasoner.

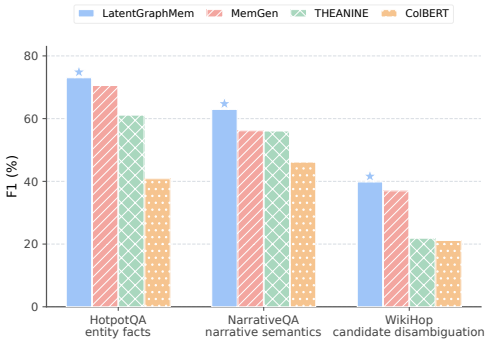

Figure 1: Dataset-specific F1 with the Qwen3-8B frozen reasoner. LATENTGRAPHMEM, MemGen, THEANINE, and ColBERT (Khattab and Zaharia, 2020) represent ours, implicit latent memory, explicit graph memory, and text retrieval. HotpotQA, NarrativeQA, and WikiHop stress entity facts, narrative semantics, and candidate disambiguation.

1 Introduction

Long-context question answering requires a model to answer a query from sparse, noisy evidence dispersed across a long input (Bai et al., 2025; Moon and Lim, 2025). Long-chain reasoning and multi-agent workflows further require systems to accumulate, organize, and reuse information beyond a local span (Hu et al., 2025). However, full-context prompting can dilute relevant evidence, while chunk retrieval can miss relational dependencies across entities and events. Effective memory needs selective access to task-relevant structure while keeping the reasoning input compact.

Prior work has therefore developed memory mechanisms for large language model (LLM) systems along two main directions. On the one hand, explicit structured memories maintain inspectable external artifacts, often with graph-structured representations (Huang et al., 2025; Chhikara et al., 2025; Ong et al., 2025). Recent hierarchical and dynamic variants further improve organization through multi-level graph retrieval, temporal graph partitioning, and adaptive indexing (Rasmussen et al., 2025; Anokhin et al., 2025; Gutierrez et al., 2024). On the other hand, latent memory mechanisms store and manipulate information directly in continuous representations (Zhang et al., 2025; Wang et al., 2025b,a), which reduces prompt expansion and improves compactness during memory use.

Despite this progress, existing memory interfaces remain incomplete when evidence structures vary across tasks. Explicit graph memories still rely on discrete graph construction and symbolic retrieval: entity extraction and relation induction can introduce local errors, graph expansion can increase indexing ambiguity, and retrieved edges may

be structurally plausible yet task-irrelevant. Latent memories can retain softer semantic information, but their retrieved states are difficult to inspect or audit. Figure 1 summarizes representative F1 results under a fixed 8B frozen reasoner and illustrates that the bottleneck changes with the evidence structure. HotpotQA is relatively friendly to entity facts, NarrativeQA requires event and discourse semantics, and WikiHop stresses candidate and property disambiguation. Together, the patterns in Figure 1 suggest that long-context QA needs a memory interface that retrieves over robust continuous representations while presenting compact, auditable evidence to the reasoner.

To bridge this gap, we propose LatentGraphMem, a symbolic-latent memory framework with a graph-memory builder, a subgraph retriever, and a frozen reasoner. LatentGraphMem keeps symbolic triples and latent edge rows aligned throughout the pipeline. The **graph-memory builder** is distilled from document-only teacher triples and produces a bounded symbolic edge set with aligned edge embeddings. Document-only teacher supervision supplies explicit edge text, so the latent memory used later is produced by the student builder rather than copied from the teacher. After the builder is frozen, the **subgraph retriever** scores builder-induced edge embeddings with the query and selects a compact evidence subgraph under a fixed budget. Retrieval is trained in latent edge space while preserving a concrete edge set for inspection. The **frozen reasoner** then receives a deterministic serialization of the selected triples, allowing the same memory interface to support answer generation across different reasoner scales. Our contributions are as follows:

- We co-produce explicit triples and row-aligned latent edge representations, learn retrieval over the latent rows, and condition a frozen downstream LLM on selected explicit triples.
- We distill and align a graph-memory builder from document-only teacher triples, then train an answer-grounded retriever over frozen builder-induced edge tables.
- We evaluate on HotpotQA, NarrativeQA, and WikiHop across three frozen-reasoner scales. LatentGraphMem gives strong macro-average EM, F1, and ROUGE-1 results, and ablations isolate the roles of latent retrieval and explicit evidence exposure.

2 Related Work

2.1 Explicit Structured Memory and Graph-Based Retrieval

A dominant line of agent memory research builds an *explicit* external store and retrieves from it at test time. Early and flat designs include text, key-value, or database-style memories such as MemGPT (Packer et al., 2023). More recent systems impose relational or temporal structure to support controlled retrieval and updates, including graph variants of production-oriented long-term memory (Mem0 / Mem0g) (Chhikara et al., 2025) and dynamic indexing/linking designs such as A-Mem (Xu et al., 2025).

Graph-based retrieval further organizes memory around higher-level structure. GraphRAG (Edge et al., 2024) constructs a multi-level graph index via community detection and recursive summarization, while Zep (Rasmussen et al., 2025) models agent memory as a temporal knowledge graph with community partitioning. THEANINE (Ong et al., 2025) retrieves coherent temporal trajectories for long-term dialogue, and PREMem (Kim et al., 2025) improves structured memory construction through pre-storage reasoning. Explicit-structure approaches are interpretable, but their end-to-end performance is sensitive to extraction, indexing, and retrieval errors. Their structured artifacts must also be serialized back 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

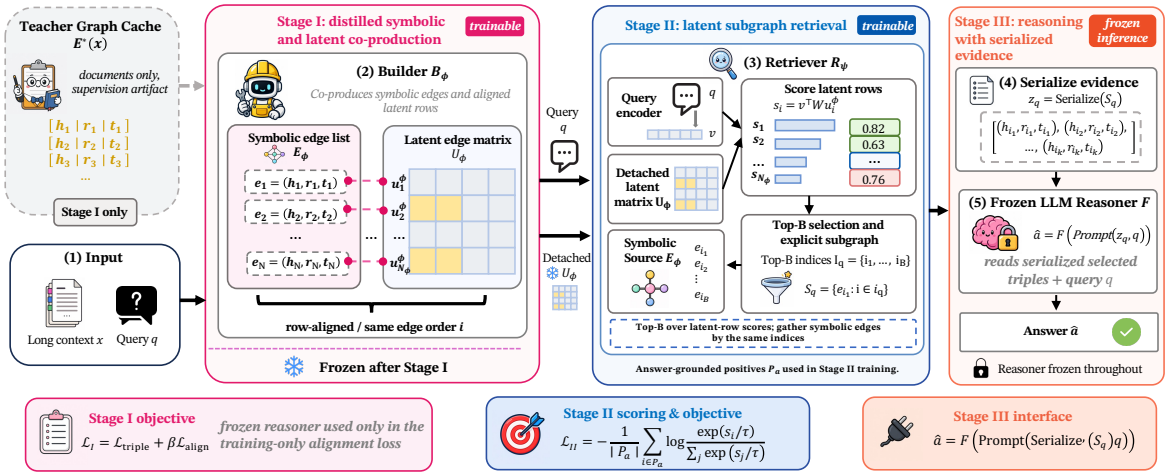

Figure 2: **Overview of LatentGraphMem.** A builder induces a bounded symbolic edge view with aligned latent rows, a retriever selects a query-conditioned subgraph in latent space, and a frozen reasoner answers from serialized selected triples. A lower panel summarizes the training objectives and inference interface.

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

LatentGraphMem learns a symbolic-latent memory interface for long-context question answering. Given a document x and a query q, the framework maintains an auditable symbolic edge view and aligned continuous edge embeddings. LatentGraphMem contains three modules: a **graph-memory builder** B_ϕ, a **subgraph retriever** R_ψ, and a **frozen LLM reasoner** F. A **document-only teacher cache** provides pseudo-labels for distilling the builder. After this distillation step, retrieval is defined over the memory induced by the trained builder rather than over the teacher cache itself. Retriever training uses **frozen builder-induced edge embeddings**, while the reasoner reads a deterministic serialization of the selected symbolic triples. We never fine-tune the reasoner.

Separating teacher supervision, builder induction, and reasoning keeps the implementation faithful to the symbolic-latent interface. Teacher cache entries supply only explicit edge text and never latent vectors. A student builder converts bracketed edges into **row-aligned latent edge embeddings**. Builder-induced tables can be materialized as frozen tensors for retriever training without changing the objective. At inference time, selected indices couple latent retrieval scores to serialized symbolic evidence, so the default reasoning interface does not require injecting memory embeddings into a particular frozen reasoner.

Notation and reasoner interface. A training example is (x, q, a), where $a = (a_1, \ldots, a_{|a|})$ is the gold answer. A graph view is a bounded edge set $\mathcal{E} = \{e_i\}_{i=1}^N$ with $e_i = (h_i, r_i, t_i)$. Its aligned latent edge matrix is $U = [u_1, \ldots, u_N] \in \mathbb{R}^{N \times d}$, where row u_i corresponds to edge e_i. For an edge subset S, the reasoner receives a serialized evidence string

$$\begin{aligned} z_S &= \text{Serialize}(S) \\ &= \text{Bracket}\big(\{(h_i, r_i, t_i) : e_i \in S\}\big), \end{aligned} \tag{1}$$

where $\text{Bracket}(\cdot)$ uses the same triple syntax for teacher supervision and builder outputs. With $E_F(\cdot)$ denoting the token embedding lookup of F, the default reasoner input is

$$X_S = E_F(\text{Prompt}(z_S, q)), \tag{2}$$

and the reasoner decodes the answer from this text prompt. Latent rows in U are used for retrieval, while row-aligned symbolic triples in S are used for evidence presentation. Algorithm 1 in Appendix B.2 summarizes the default training and inference flow.

3.1 Stage I: Distilled Symbolic and Latent Co-production

Stage I trains the builder to co-produce bracketed symbolic triples and aligned latent edge embed-

dings. Before builder distillation, we construct a teacher graph cache from the document x only. Teacher extraction does not observe the query or answer. Cache entries store pseudo-gold triples $\mathcal{E}^\star$ and record extractor provenance. Teacher extraction supplies only explicit edge text, not hidden states or latent vectors. We use the cache as a supervision artifact. It fixes the symbolic target for Stage I, but it is not the memory object optimized by the retriever. Each edge list is written in a bracketed format,

$$\begin{aligned} y_{\mathcal{E}^\star} &= \mathrm{Bracket}(\mathcal{E}^\star) \\ &= \texttt{[}h_1 \ \texttt{|}\ r_1 \ \texttt{|}\ t_1\texttt{]} \cdots \\ &\quad \texttt{[}h_N \ \texttt{|}\ r_N \ \texttt{|}\ t_N\texttt{]}, \end{aligned} \tag{3}$$

with one edge per line. Teacher cache construction and parsing are discrete preprocessing operations. Differentiable builder training begins after the teacher target is loaded.

Given x and $y_{\mathcal{E}^\star}$, the builder performs one teacher-forced forward pass,

$$(H, \ell) = B_\phi(x, y_{\mathcal{E}^\star}), \tag{4}$$

where ℓ are LM-head logits on the bracket target and H are hidden states. That forward pass also produces the edge matrix. For each teacher edge, we locate the tail field t_i in the bracket target, map its character span to token positions, pool the corresponding hidden states, and project:

$$\begin{aligned} g_i &= \mathrm{Pool}\big(H_{\mathrm{span}(t_i)}\big), \\ u_i &= P_{\mathrm{edge}} g_i. \end{aligned} \tag{5}$$

We reuse the same bracketed format and tail-span pooling when the trained builder embeds a bracketed edge list generated by the builder. Thus every student symbolic edge remains row-aligned with one latent row, and the conversion from explicit edges to latent rows is performed by the student builder rather than by the teacher.

Stage I uses two losses. A token-level triple loss distills bracketed edge generation:

$$\mathcal{L}_{\mathrm{triple}} = -\sum_{j \in \mathcal{Y}} \log p_\phi(y_j \mid y_{<j}, x). \tag{6}$$

Here $\mathcal{Y}$ indexes the bracket target tokens. Triple loss directly trains symbolic triple generation. It does not by itself train the projection P_{edge} in Eq. 5. To train the latent rows used by the retriever, Stage I also applies a training-only alignment loss through the frozen reasoner. Let $U^\star = [u_1, \ldots, u_N]$ be the latent rows produced from the teacher edge list. Let P_F be a training-only adapter that maps edge rows into the reasoner's embedding space. It is the identity map when dimensions match and a learned linear projection otherwise. Let $\widetilde{U}^\star = [P_F u_i]_{i=1}^N$ denote the row-wise projected edge matrix. With F frozen,

$$\mathcal{L}_{\mathrm{align}} = -\sum_{t=1}^{|a|} \log p_F\Big(a_t \mid a_{<t},\, \widetilde{U}^\star \oplus E_F(q)\Big). \tag{7}$$

We optimize the Stage I objective

$$\mathcal{L}_{\mathrm{I}} = \mathcal{L}_{\mathrm{triple}} + \beta\, \mathcal{L}_{\mathrm{align}}. \tag{8}$$

Gradients from $\mathcal{L}_{\mathrm{align}}$ update P_{edge}, the builder adapter parameters, and P_F when it is non-identity and trainable. Reasoner parameters remain frozen. Questions and answers enter only this post-extraction alignment objective. They are not used to construct or filter the teacher cache. Alignment only makes the row-aligned latent edge representations useful for retrieval. Default inference still passes serialized symbolic triples to the reasoner.

3.2 Stage II: Latent Subgraph Retrieval

Stage II freezes the distilled builder and trains the retriever on the student symbolic-latent edge table induced by that builder. Conceptually, the builder first produces a bounded symbolic graph view

$$\mathcal{E}_\phi(x) = B_\phi^{\mathrm{gen}}(x) = \{e_i\}_{i=1}^{N_\phi}, \tag{9}$$

and then embeds the same ordered edge list with its frozen edge-embedding interface:

$$\begin{aligned} U_\phi &= B_\phi^{\mathrm{embed}}(\mathcal{E}_\phi), \\ \mathrm{stopgrad}(U_\phi) &= U_\phi. \end{aligned} \tag{10}$$

Here $U_\phi = [u_1^\phi, \ldots, u_{N_\phi}^\phi]$ is produced by the frozen builder from the same ordered student edge list $\mathcal{E}_\phi$. For efficiency, this builder-induced table can be materialized once and reused during retriever training. Such a materialized student-memory table is a frozen snapshot of B_ϕ outputs, not the teacher cache $\mathcal{E}^\star$ used for Stage I supervision. Materialization does not change the training objective and no gradients pass into the builder. Retriever scoring encodes the question, applies a query projection, and scores every edge by a bilinear form,

$$\begin{aligned} h_q &= R_\psi^{\mathrm{enc}}(q), \\ v &= P_q h_q, \\ s_i &= v^\top W u_i^\phi. \end{aligned} \tag{11}$$

Given retrieval budget B, forward selection uses hard Top-B indices $\mathcal{I}_q = \mathrm{Top}_B(s)$, yielding the auditable subgraph $S_q = \{e_i : i \in \mathcal{I}_q\}$. Identical indices select latent rows for scoring and symbolic triples for downstream reasoning.

For retriever supervision, we derive answer-grounded pseudo-positive edges from the symbolic edge table:

$$\mathcal{P}_a = \{\, i : \mathrm{Normalize}(a) \subseteq \mathrm{Normalize}(h_i, r_i, t_i), \; e_i \in \mathcal{E}_\phi \,\}. \tag{12}$$

Here $\mathrm{Normalize}(\cdot)$ lowercases text and removes punctuation before matching the answer against the concatenated edge text. Examples with $\mathcal{P}_a = \emptyset$ are skipped for retriever training. We train with the Stage II ranking loss

$$\mathcal{L}_{\mathrm{II}} = -\frac{1}{|\mathcal{P}_a|} \sum_{i \in \mathcal{P}_a} \log \frac{\exp(s_i/\tau)}{\sum_{j=1}^{N_\phi} \exp(s_j/\tau)}. \tag{13}$$

Since U_ϕ is detached, Stage II updates the retriever query encoder, query projection P_q, and W, but not the builder or the reasoner.

3.3 Stage III: Reasoning with Serialized Evidence

After latent retrieval, the selected indices define $S_q = \{e_i : i \in \mathcal{I}_q\}$. Selected symbolic triples are serialized with the same bracketed format used by the builder:

$$z_q = \mathrm{Serialize}(S_q). \tag{14}$$

A frozen reasoner then generates the answer from the question and serialized evidence:

$$\hat{a} = F(\mathrm{Prompt}(z_q, q)). \tag{15}$$

Serialized-evidence reasoning uses explicit evidence presentation after latent retrieval. No builder or retriever parameters are updated during this step.

4 Experiments

4.1 Datasets and Baselines

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)). We evaluate on a fixed 2,600-instance test mixture drawn from HotpotQA (Yang et al., 2018), NarrativeQA (Kociský et al., 2018), and WikiHop (Welbl et al., 2018). Table 2 reports statistics computed on our processed splits, including character counts and token counts under the reporting tokenizer for consistent measurement across configurations. Additional statistics and dataset characteristics are provided in Appendix A.

Baselines. We compare LatentGraphMem with a reasoner-only baseline and seven retrieval or memory-enabled baselines. Reasoner-only (full context) answers directly from the full input (up to the context limit) without any external memory. ColBERT (Khattab and Zaharia, 2020), LightRAG (Guo et al., 2025), and RAG (Text) retrieve textual evidence and append it to the same frozen reasoner. We additionally include representative memory systems. For explicit structure, we evaluate THEANINE (timeline-style graph memory management) (Ong et al., 2025), PREMem (pre-storage reasoning for episodic memory) (Kim et al., 2025), and A-Mem (agentic memory with dynamic indexing and linking) (Xu et al., 2025). For latent memory, we evaluate MemGen (generative latent memory tokens) (Zhang et al., 2025). All baselines are instantiated as memory or retrieval modules over the same static document-grounded QA inputs, so the comparison focuses on evidence selection and answer generation rather than interactive memory update.

4.2 Implementation Details

For all experiments, the builder and retriever are decoder-only LLM backbones with parameter-efficient LoRA adapters, while the downstream reasoner remains frozen. Unless otherwise specified, the builder and retriever use Qwen2.5-1.5B-Instruct, and the frozen reasoner is varied across Qwen2.5-1.5B-Instruct, SmolLM3-3B, and Qwen3-8B to test scale transfer. Stage I distills the builder from a versioned Qwen2.5-32B teacher graph cache constructed from documents only, without query, answer, or supporting-fact access. Stage II freezes the builder and trains the retriever on builder-induced symbolic-latent edge tables. The Stage I alignment reasoner and P_F are training-only, while inference uses serialized triples for every evaluated reasoner. Long documents are processed in 1,024-token chunks with 128-token overlap. Extraction is capped at 32 triples per chunk, and global graph capacity is limited to $M = 150$ edges. Retrieved evi-

Table 1: Main QA results. We report EM/F1/ROUGE-1 (R1) and macro averages over datasets. Values are percentages. Bold dark-blue values denote the best score per column. Pale yellow denotes second-best. The LATENTGRAPHMEM row is lightly shaded.

Method	HotpotQA			NarrativeQA			WikiHop			Avg		
	EM	F1	R1	EM	F1	R1	EM	F1	R1	EM	F1	R1
Backbone: Qwen2.5-1.5B												
Reasoner-only	2.60	19.35	19.75	2.25	27.36	29.50	0.75	7.85	7.71	1.87	18.19	18.99
ColBERT	32.30	44.21	46.20	27.00	51.38	53.75	15.62	28.10	29.50	24.97	41.23	43.15
LightRAG	46.70	52.47	53.78	1.38	5.56	7.26	4.00	8.89	10.30	17.36	22.31	23.78
A-Mem	47.60	54.86	56.34	23.38	47.19	49.51	12.25	19.98	20.83	27.74	40.68	42.23
MemGen	50.70	57.29	58.19	23.50	40.26	41.60	18.00	28.38	29.19	30.73	41.98	42.99
RAG	14.70	21.05	21.29	4.50	19.41	20.30	1.50	6.38	6.23	6.90	15.61	15.94
PREMem	45.60	51.28	52.96	6.38	13.15	14.33	5.38	10.52	11.31	19.12	24.98	26.20
THEANINE	51.20	57.34	58.37	25.87	48.89	50.89	13.63	21.91	22.69	30.23	42.71	43.98
LATENTGRAPHMEM	**54.60**	**60.67**	**61.28**	**28.38**	**53.56**	**54.80**	**21.50**	**32.15**	**32.36**	**34.83**	**48.79**	**49.48**
Backbone: SmolLM3-3B												
Reasoner-only	45.50	57.16	59.08	27.75	53.50	55.64	16.75	29.09	30.15	30.00	46.58	48.29
ColBERT	36.20	48.63	50.10	13.00	31.71	33.89	12.62	21.39	22.23	20.61	33.91	35.41
LightRAG	44.20	53.19	54.03	1.38	7.32	9.40	4.38	10.91	12.27	16.65	23.81	25.23
A-Mem	**52.30**	59.30	61.53	30.38	55.22	56.95	17.38	27.49	28.88	33.35	47.34	49.12
MemGen	50.50	57.04	58.66	24.88	48.38	49.41	24.12	30.47	31.29	33.17	45.30	46.45
RAG	16.80	27.41	27.90	6.88	31.74	33.31	2.25	10.31	10.10	8.64	23.15	23.77
PREMem	46.33	58.68	59.63	6.81	14.71	16.63	17.31	26.26	27.60	23.48	33.22	34.62
THEANINE	14.03	29.00	29.35	16.62	42.10	43.85	6.88	15.07	15.55	12.51	28.72	29.58
LATENTGRAPHMEM	50.50	**61.60**	**62.15**	**31.87**	**58.69**	**60.37**	**30.77**	**41.38**	**43.17**	**37.71**	**53.89**	**55.23**
Backbone: Qwen3-8B												
Reasoner-only	3.80	25.69	27.29	8.75	39.62	41.56	1.00	11.87	11.32	4.52	25.73	26.72
ColBERT	21.60	40.84	43.31	16.38	46.15	48.61	9.75	21.15	21.47	15.91	36.05	37.80
LightRAG	42.90	51.00	53.46	2.25	8.22	10.77	4.00	9.72	10.63	16.38	22.98	24.95
A-Mem	34.40	44.11	44.85	25.50	44.93	46.69	9.62	16.65	16.95	23.17	35.23	36.16
MemGen	63.50	70.56	72.76	27.75	56.14	58.10	24.88	36.98	37.80	38.71	54.56	56.22
RAG	29.30	39.88	40.62	15.00	38.17	39.93	5.12	13.13	13.12	16.47	30.39	31.22
PREMem	52.00	60.18	61.96	1.00	7.70	9.89	16.62	27.99	29.24	23.21	31.96	33.70
THEANINE	52.65	61.11	62.91	29.12	56.05	58.33	11.89	21.86	22.47	31.22	46.34	47.90
LATENTGRAPHMEM	**63.60**	**73.03**	**75.08**	**35.75**	**62.95**	**64.74**	**31.37**	**39.81**	**41.01**	**43.57**	**58.60**	**60.28**

dence is serialized as bracketed triples and appended to the frozen reasoner's prompt. Additional cache, loss, and materialization details appear in Appendix B. All retrieval-based baselines use the same frozen-reasoner answer-generation protocol and context-limit handling, while LATENTGRAPHMEM exposes a fixed-budget symbolic subgraph.

We report exact match (EM), token-level F1, and ROUGE-1 (R1). For reproducibility, evaluations use fixed processed splits, a fixed graph-cache schema, and $B = 30$. After development-stage configuration selection, final teacher-cache generation and answer decoding use deterministic settings across methods. We set temperature to 0, disable sampling, use top-$p = 1$, fix seeds where applicable, and score all systems with matched-example EM/F1/R1 scripts.

4.3 Main Results

Macro averages and transfer. Table 1 compares LATENTGRAPHMEM with eight baselines on HotpotQA, NarrativeQA, and WikiHop under Qwen2.5-1.5B, SmolLM3-3B, and Qwen3-8B frozen reasoners. Macro-average columns show that LATENTGRAPHMEM remains competitive across EM, F1, and R1 for all three backbones. We treat per-cell highlighting as descriptive and focus aggregate claims on macro averages, F1/R1 trends, and the structural analyses below. Builder and retriever parameters are trained once with Qwen2.5-1.5B and reused unchanged across reasoner scales, supporting the modularity of the serialized-evidence interface. F1 and R1 gains indicate that selected symbolic evidence improves token-level answer coverage rather than only exact-match behavior.

Dataset-level trends. Dataset-level results support the motivation that long-horizon QA is not a single scaling problem. HotpotQA is often entity-centric and triple-friendly, so several retrieval and graph baselines are competitive, but LATENTGRAPHMEM still achieves the best F1 and

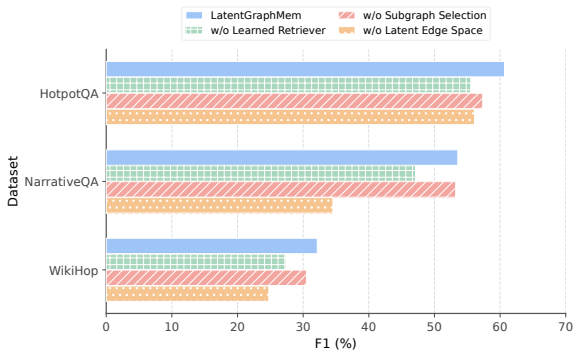

Figure 3: Structural ablation on dataset-level F1 with the 1.5B backbone. Variants remove the learned retriever, budgeted subgraph selection, or latent edge space.

R1 at every backbone scale. NarrativeQA requires event, coreference, and causal context, so methods differ more sharply in how they balance broad semantic coverage with answer-focused evidence selection. WikiHop stresses candidate and property disambiguation rather than simply including the answer string, and the gap between backbone scales shows that larger reasoners do not automatically resolve retrieval-time ambiguity. LATENTGRAPHMEM remains competitive across these different evidence structures.

Baseline failure modes. Baseline results reveal complementary failure modes. Text retrieval methods such as ColBERT can work when retrieved passages contain the answer-bearing span, but their performance varies sharply across datasets and backbones. LightRAG provides broad coverage, while the supporting-fact analysis in Appendix D shows that high recall can come with low precision and distractors. MemGen is the strongest latent-memory baseline but does not expose inspectable evidence. Explicit-memory systems such as A-Mem and THEANINE achieve isolated strong results, but they do not maintain the same consistency across entity facts, narrative semantics, and candidate-heavy reasoning.

4.4 Structural Ablation

We conduct structural ablations to isolate the contribution of each memory component in LATENTGRAPHMEM, as shown in Figure 3. Our full model achieves the best F1 on all three datasets, showing that the symbolic graph, latent edge space, learned retriever, and budgeted subgraph selection are complementary. Removing the learned retriever consistently reduces performance, indicating that heuristic edge selection is less effective at identifying task-relevant evidence. Removing subgraph selection also hurts performance, suggesting that exposing the full graph introduces noise to the frozen reasoner. Removing the latent edge space causes the largest drop, especially on NarrativeQA and WikiHop, confirming that latent edge representations are important for robust retrieval under dispersed evidence.

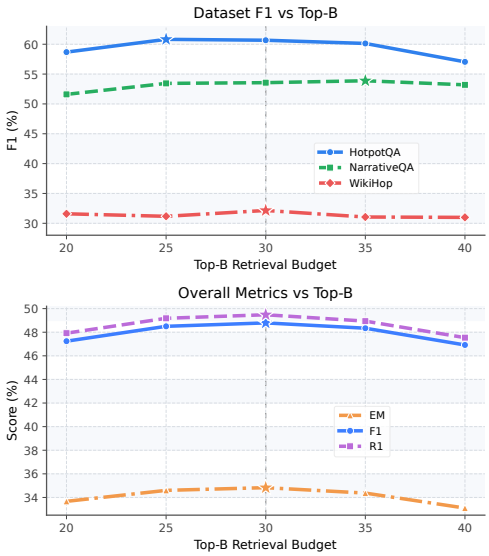

Figure 4: Retrieval-budget sensitivity with the 1.5B backbone. Top: dataset F1. Bottom: overall EM, F1, and R1. Stars mark per-curve best budgets and the dashed line marks $B = 30$.

4.5 Effect of Retrieval Budget

We further vary the Top-B retrieval budget for LATENTGRAPHMEM with the 1.5B backbone, as shown in Figure 4. Top panel reports dataset-level F1, while the bottom panel summarizes the overall EM, F1, and ROUGE-1 scores over the evaluation mixture. Best budget differs across individual datasets: HotpotQA reaches its highest F1 at $B = 25$, NarrativeQA peaks at $B = 35$, and WikiHop performs best at $B = 30$. Budget variation reflects the different evidence structures of the benchmarks. HotpotQA benefits from a compact set of entity-centric evidence, so adding more retrieved edges can introduce distracting facts. NarrativeQA prefers a slightly larger budget, suggesting that narrative questions require broader event-level context. WikiHop is most effective in the middle range, where the selected subgraph contains enough property evidence for candidate disambiguation without admitting excessive distractors.

Despite these dataset-specific optima, the over-

all metrics favor the default setting $B = 30$. At this budget, LATENTGRAPHMEM obtains the best overall EM, F1, and ROUGE-1 scores, while also achieving the best WikiHop F1. Moreover, $B = 30$ remains close to the dataset-specific best settings on the other two benchmarks: it is only 0.15 F1 below the HotpotQA peak at $B = 25$ and 0.32 F1 below the NarrativeQA peak at $B = 35$. We therefore use $B = 30$ as the default retrieval budget, as it provides the most balanced trade-off between evidence coverage and retrieval noise across datasets and metrics. Appendix D.1 reports evidence-compression statistics for the selected subgraphs.

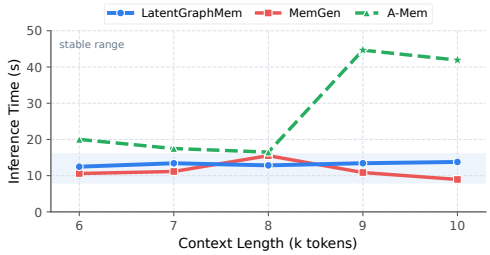

Figure 5: Inference time for long-context samples (≥6k). LATENTGRAPHMEM stays stable as context length grows, while A-Mem rises at longer contexts.

4.6 Efficiency and Qualitative Analysis

Inference scalability. We evaluate the inference-time behavior of different memory systems under increasing context lengths to assess 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 Figure 5, LATENTGRAPHMEM remains within a narrow latency band across all evaluated context lengths, with inference time staying around 12 to 14 seconds from 6k to 10k tokens. MemGen is also efficient, while LATENTGRAPHMEM provides comparable efficiency with more stable latency across context lengths. In contrast, A-Mem exhibits unstable scaling behavior: although its runtime is moderate from 6k to 8k tokens, it increases sharply at longer contexts, reaching 44.65 seconds at 9k tokens and remaining high at 10k tokens. Latency patterns suggest that latent edge retrieval with a fixed evidence budget effectively limits reasoner-side cost growth, even though the selected evidence is ultimately exposed to the frozen reasoner as explicit triples.

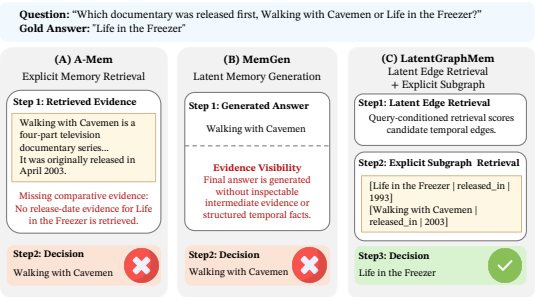

Figure 6: Case study on temporal evidence. A-Mem retrieves one side, MemGen exposes no intermediate evidence, and LATENTGRAPHMEM retrieves both release dates and answers correctly.

Case study. Figure 6 examines a temporal comparison question involving *Walking with Cavemen* and *Life in the Freezer*, where relevant release dates are dispersed across a long context. Solving the question requires evidence for both entities, not only a salient date mention. A-Mem retrieves only one side and answers incorrectly, while MemGen also outputs *Walking with Cavemen* without inspectable intermediate evidence. In contrast, LATENTGRAPHMEM retrieves explicit release-date edges for both entities, allowing the frozen reasoner to compare the dates and answer *Life in the Freezer*. Selected triples expose the comparison path used by the reasoner, making the answer trace easier to audit. Overall, the case illustrates how latent edge retrieval can select a compact symbolic subgraph that preserves the evidence needed for multi-hop comparison.

5 Conclusion

We present LATENTGRAPHMEM, a memory framework that integrates latent edge retrieval with explicit, budgeted subgraph reasoning. Row-aligned symbolic triples and latent edge embeddings combine stable retrieval with inspectable evidence exposure. Experiments across long-horizon benchmarks show strong performance, stable inference time, and transfer to larger frozen reasoners, highlighting the value of symbolic-latent memory for reasoning under long and dispersed contexts. Results also show that a builder-retriever memory trained once can work with larger frozen reasoners through serialized evidence. Together, these findings support row-aligned symbolic-latent memory as a practical route to compact retrieval with auditable reasoning traces.

Limitations

While LATENTGRAPHMEM targets long-horizon reasoning with sparse evidence, our current evaluation focuses on static document-grounded text question answering. Extending the same row-aligned symbolic-latent interface to multimodal evidence, streaming contexts, and interactive memory-update settings remains future work. Fixed graph capacities and retrieval budgets improve comparability and stability, but adaptive budget selection may further improve efficiency across tasks with different evidence structures. The teacher cache is built with one document-only extractor schema; exploring alternative teacher extractors is orthogonal to the proposed builder-retriever interface.

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.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. Lightrag: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 10746–10761. Association for Computational Linguistics.

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*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48. ACM.

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.

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 Dataset Statistics and Characteristics

Table 2 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 reporting tokenizer to ensure comparability across dataset rows.

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.

Dataset	Split	#Inst.	Context avg.		Answer avg.	
			Chars	Tokens	Chars	Tokens
Training corpora						
TriviaQA (Joshi et al., 2017)	Train	14,679	75,314	18,828	9.9	2.1
QASPER (Dasigi et al., 2021)	Train	3,598	24,608	6,152	6.0	1.0
QuALITY (Pang et al., 2022)	Train	2,523	27,018	6,754	49.4	12.0
Evaluation corpora						
HotpotQA (Yang et al., 2018)	Test	1,000	5,240	1,310	11.0	2.7
NarrativeQA (Kociský et al., 2018)	Test	800	3,860	965	26.0	6.5
WikiHop (Welbl et al., 2018)	Test	800	7,230	1,807	12.3	3.1
Evaluation mixture†	Test	2,600	5,428	1,356	16.0	4.0

Table 2: Statistics of the training and evaluation datasets. Context and answer lengths are reported as average character counts and token counts under the reporting tokenizer. † denotes averages over the fixed evaluation mixture.

Evaluation datasets. **HotpotQA** (Yang et al., 2018) and **WikiHop** (Welbl et al., 2018) are multi-hop benchmarks requiring reasoning across multiple 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. Table 2 reports length statistics for the *Evaluation mixture* row averaged over this mixture, consistent with our main evaluation protocol.

B Implementation Details and Prompts

Stage I uses a versioned teacher graph cache as pseudo-gold supervision. In the main run, this cache is produced by a Qwen2.5-32B teacher from documents only, without access to the query, answer, or supporting facts, under the same bracketed triple schema used by the builder. Teacher extraction provides only explicit bracketed triples. It does not provide hidden states or latent vectors.

Builder distillation co-produces symbolic and latent evidence. A teacher-forced LM loss trains bracketed triple generation. Tail-span hidden states from the same forward pass are pooled and projected by P_{edge} to produce edge embeddings. Stage I optimizes $\mathcal{L}_{\text{triple}} + \beta\mathcal{L}_{\text{align}}$. Alignment uses a training-only frozen-reasoner loss. Its gradients update P_{edge}, the builder adapters, and P_F when P_F is learned. Reasoner parameters remain frozen.

After distillation, Stage II freezes the builder and trains the retriever on the builder-induced symbolic-latent edge table. For efficiency, frozen builder outputs are materialized before retriever training. Materialized tables are student-memory snapshots rather than the teacher cache used for Stage I supervision. Edges whose normalized builder-produced text contains the normalized answer are treated as pseudo-positive retrieval targets. Examples with no such edge are skipped for retriever supervision. Alignment reasoner calls and P_F are used only inside the training objective. Inference uses serialized text evidence, so evaluation reasoners can be swapped without changing the builder-retriever interface.

B.1 Prompt Templates

Teacher extractor and student builder use the same bracketed extraction instruction for teacher-cache construction and live extraction. During distillation, the student is teacher-forced against the bracketed target. Retriever scoring is not instruction-prompted. It encodes the raw question and scores latent edge rows. After retrieval, selected symbolic edges are serialized into the answer prompt read by the frozen reasoner.

B.2 Default Training and Inference Flow

Algorithm 1 summarizes the default training and inference pipeline. Stage I builds a document-only teacher cache and distills the builder with teacher-forced bracket triples plus the training-only latent alignment loss. Stage II freezes the builder, materializes the student symbolic-latent edge table, and trains the retriever on answer-grounded positives from that table. At inference time, the retriever scores frozen builder-induced edge rows, selects

Stage I: Builder extraction target

Used for: teacher-forced builder distillation.

Natural-language extraction template

```
Extract knowledge graph triples from
the text.
Output one triple per line in the
exact format:
[head | relation | tail]
Text:
{document chunk}
Triples:
```

Target format:

$$y_{E^\star} = \mathrm{Bracket}(E^\star) = [h_1|r_1|t_1]\cdots[h_N|r_N|t_N].$$

Stage II: Retriever scoring interface

Used for: latent subgraph retrieval.

Note: interface inputs here are not text prompts. Retriever scoring uses detached builder-induced edge rows.

$$\begin{aligned} h_q &= R_\psi^{\mathrm{enc}}(q), \\ v &= P_q h_q, \\ s_i &= v^\top W u_i^\phi, \\ I_q &= \mathrm{Top}_B(s), \\ S_q &= \{e_i : i \in I_q\}. \end{aligned}$$

Training signal:

$$P_a = \{i : \mathrm{Norm}(a) \subseteq \mathrm{Norm}(h_i, r_i, t_i), \; e_i \in E_\phi\}.$$

Stage III: Frozen reasoner prompt

Used for: answer generation from selected explicit evidence.

Serialized evidence:

$$z_q = \mathrm{Serialize}(S_q).$$

```
Answer the question based on the
given context.
Provide a clear and concise answer.
Context: Evidence triples:
[head | relation | tail]
...
Question: {question}
Answer:
```

Frozen reasoner:

$$\hat{a} = F(\mathrm{Prompt}(z_q, q)).$$

Figure 7: Prompt and interface templates used by LATENTGRAPHMEM. Stage I uses a bracketed triple extraction target for builder distillation. Stage II trains the retriever through a latent scoring interface over detached builder-induced edge rows. Stage III gives the frozen reasoner serialized selected triples as text evidence.

Algorithm 1 Default training and inference flow of LATENTGRAPHMEM.

Require: Training examples (x, q, a), document-only teacher extractor, frozen reasoner F, retrieval budget B

1: Build a versioned teacher cache $\mathcal{E}^\star(x)$ from documents x only.
2: Distill and align builder B_ϕ with teacher-forced bracket triples and a training-only latent alignment loss.
3: Freeze B_ϕ and induce a student-memory table $\mathcal{E}_\phi(x) = B_\phi^{\mathrm{gen}}(x)$, $U_\phi = B_\phi^{\mathrm{embed}}(\mathcal{E}_\phi)$.
4: Train retriever R_ψ on detached U_ϕ using answer-grounded positives from $\mathcal{E}_\phi$.
5: At inference, score $s_i = v^\top W u_i^\phi$, select $\mathcal{I}_q = \mathrm{Top}_B(s)$, and serialize $S_q = \{e_i : i \in \mathcal{I}_q\}$.
6: Generate $\hat{a} = F(\mathrm{Prompt}(\mathrm{Serialize}(S_q), q))$ with F frozen.

a Top-B subgraph, and passes only the serialized selected triples to the frozen reasoner.

C Reproducibility Protocol

All reported systems are evaluated on the same processed example identifiers and scored with the same EM/F1/ROUGE-1 implementation. After development-stage configuration selection, teacher-cache generation and frozen-reasoner answer decoding use the same fixed deterministic settings across methods: temperature 0, no sampling, and top-$p = 1$. Where preprocessing, materialization, or retriever training involves stochastic operations, we use fixed random seeds and keep the graph-cache schema, retrieval budget, and evaluation split fixed.

D Additional Efficiency and Evidence Analysis

D.1 Token and Evidence Compression

Table 3 reports the evidence payload read by the frozen reasoner after subgraph selection. Overall, LATENTGRAPHMEM selects 28.66 edges on average and serializes them into 421.96 tokens, compared with 825.34 tokens for the full builder-induced graph. Thus, the reasoner reads a compact selected subgraph rather than the entire graph or the original long context. Compression is dataset dependent. NarrativeQA has the shortest selected graph in absolute tokens, but its selected/context ratio is the highest because its contexts are shorter after preprocessing. WikiHop has the largest full-graph token tail, reflecting many candidate entities and properties, while Top-B selection keeps the reasoner payload close to 439 tokens on average. Together with the inference-time results, this report shows that budgeted latent retrieval controls the text evidence exposed to the frozen reasoner.

Latent memory footprint is also bounded by the global graph capacity. In our implementation, $d = 1536$ and $M = 150$, so the builder-induced latent edge matrix U_ϕ contains at most $M \times d = 230{,}400$ values per example. This corresponds to approximately 0.45 MB in bfloat16 and 0.9 MB in float32. For the bilinear scorer in Eq. 11, scoring all edges can compute $W^\top v$ once and then match it against at most M rows. The resulting cost is $O(d^2 + Md)$ for scoring and $O(M \log B)$ for Top-B selection. Since M and B are fixed

Table 3: Evidence compression statistics for selected and full builder-induced graphs. Token statistics are reported as mean, median, and p95. Selected graph tokens measure the serialized evidence passed to the reasoner after Top-B retrieval, while full graph tokens measure the serialized builder-induced graph before subgraph selection. Reduction is computed against the full graph token mean. Selected/context is averaged over per-example selected/context token ratios.

Source	Selected edges	Selected graph tokens			Full graph tokens			Reduction vs. full (%)	Selected/context (%)
		Mean	Median	P95	Mean	Median	P95		
Overall	28.66	421.96	432.0	550.0	825.34	771.0	1577.05	48.9	40.81
HotpotQA	28.89	449.15	454.5	567.05	752.38	808.5	1056.0	40.3	37.42
NarrativeQA	27.73	371.20	392.0	492.05	650.03	729.5	823.0	42.9	46.00
WikiHop	29.30	438.73	440.0	564.05	1091.86	802.0	2290.35	59.8	39.84

Table 4: Answer-bearing edge coverage in builder-induced graphs and selected subgraphs. Values are percentages. Drop denotes the absolute coverage decrease from the full graph to the selected subgraph, and retriever retention is computed among examples whose full graph contains an answer-bearing edge.

Source	Full graph	Selected subgraph	Drop	Retriever retention
Overall	**50.92**	**48.65**	**2.27**	**95.54**
HotpotQA	48.30	47.00	1.30	97.31
NarrativeQA	35.00	33.38	1.62	95.36
WikiHop	70.12	66.00	4.12	94.12

in the evaluation protocol, query-time retrieval is bounded by memory capacity rather than by the original document length.

D.2 Answer-Bearing Edge Coverage

Table 4 evaluates whether the retrieved symbolic subgraph preserves answer-bearing evidence under the same normalized text-matching criterion used for retriever supervision. Full builder-induced graphs contain at least one answer-bearing edge in 50.92% of all examples, and selected subgraphs contain such an edge in 48.65% of all examples. More importantly, when the full graph contains an answer-bearing edge, Top-B retrieval keeps one in the selected subgraph 95.54% of the time. High conditional coverage indicates that the latent retriever usually preserves answer-relevant symbolic evidence when the builder has produced it. Conditional coverage remains high across all three datasets, with 97.31% on HotpotQA, 95.36% on NarrativeQA, and 94.12% on WikiHop. Remaining gaps are therefore more often tied to graph construction coverage, property-level ranking, or downstream answer selection than to the selected subgraph discarding available answer-bearing edges.

D.3 Supporting-Fact Evidence Alignment

To further evaluate whether the subgraph selected by LATENTGRAPHMEM is grounded in annotated evidence, we conduct an entity-token alignment analysis on HotpotQA and WikiHop examples with gold supporting facts. For each sample, we normalize tokens from the supporting facts. We then compare them with tokens from three evidence sources: the subgraph selected by LATENTGRAPHMEM, the full builder-induced graph, and LightRAG retrieval. Table 5 reports recall, precision, and F1 over 600 stratified samples grouped by context length.

Full builder-induced graphs have higher recall but much lower precision, indicating broad coverage with substantial noise. LightRAG reaches perfect recall by retrieving broad textual evidence, but its precision decreases as contexts become longer. In contrast, the subgraph selected by LATENTGRAPHMEM obtains the best precision and F1 across all context tiers. Alignment results suggest that latent edge scoring performs selective structural filtering rather than simply exposing more context.

D.4 Supplementary Experiment Analysis

Dataset-level trends. LATENTGRAPHMEM is most effective when the builder-induced edge table preserves the sparse facts needed for compositional reasoning over dispersed evidence. It is especially strong on HotpotQA across backbones, where compact entity-centric evidence aligns well with the selected subgraph format. On NarrativeQA and WikiHop, where answers require broader event semantics or candidate-property disambiguation, methods tend to differ more in F1 and R1 than in exact-match accuracy. In these settings, LATENTGRAPHMEM keeps the retrieved evidence inspectable while relying on latent edge retrieval to

Table 5: Entity-token alignment against gold supporting facts. Values are ratios. Bold numbers indicate the best score within each tier and metric. Selected LATENTGRAPHMEM subgraphs achieve the highest precision and F1, showing that latent edge retrieval filters noisy evidence more selectively than full-graph exposure or broad chunk retrieval.

Tier	LATENTGRAPHMEM subgraph			Full graph			LightRAG		
	R	P	F1	R	P	F1	R	P	F1
Short	0.495	**0.334**	**0.330**	0.689	0.182	0.280	**1.000**	0.116	0.204
Medium	0.480	**0.342**	**0.345**	0.662	0.154	0.243	**1.000**	0.084	0.152
Long	0.440	**0.400**	**0.343**	0.657	0.145	0.230	**1.000**	0.064	0.119
Overall	0.472	**0.359**	**0.339**	0.669	0.160	0.251	**1.000**	0.088	0.159

control which symbolic facts reach the reasoner.

Failure modes of fully explicit graph-based memory. Across backbones, fully explicit graph-memory systems show high variance across datasets and metrics. PREMem is especially unstable on NarrativeQA, while A-Mem and THEANINE can be competitive in selected settings without matching the consistency of LATENTGRAPHMEM. Observed patterns suggest 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 combines this retrieval stability with explicit subgraph exposure by training over row-aligned builder-induced edge embeddings.