SAGE: A Self-Evolving Agentic Graph-Memory Engine for Structure-Aware Associative Memory

Anonymous Author(s)

Abstract

Long-term memory is becoming a central bottleneck for language agents. Existing RAG and GraphRAG systems largely treat memory graphs as static retrieval middleware, which limits their ability to recover complete evidence chains from partial cues, exploit reusable graph-structural roles, and improve the memory itself through downstream feedback. We introduce SAGE, a **S**elf-evolving **A**gentic **G**raph-memory **E**ngine that models graph memory as a dynamic long-term memory substrate. SAGE couples two roles: a memory writer that incrementally constructs structured graph memory from interaction histories, and a Graph Foundation Model-based memory reader to perform retrieval and provide feedback to the memory writer. We provide rigorous theoretical analyses supporting the effectiveness of carefully designed architectural components and the framework. Across multi-hop QA, open-domain retrieval, domain-specific review QA, and long-term agent-memory benchmarks, SAGE improves evidence recovery, answer grounding, and retrieval efficiency: after two self-evolution rounds, it achieves the best average rank on multi-hop QA; in zero-shot open-domain transfer, it reaches 82.5/91.6 Recall@2/5 on NQ. Further results on LongMemEval and HaluMem show that training and reader–writer feedback improve multiple long-term memory and hallucination-diagnostic metrics, suggesting that self-evolving, structure-aware graph memory is a promising foundation for robust long-horizon language agents. Our code is available here.

1 Introduction

As large language models evolve from single-turn question-answering systems into general-purpose agents for multi-turn dialogue, personalized assistance, multi-agent collaboration, and open-environment exploration, the system bottleneck is shifting from whether a model can answer within the current context to whether it can accumulate, organize, invoke, and update memory over longer time scales. Memory is a core system capability that determines whether Agents can achieve long-term consistency, personalized adaptation, cross-turn reasoning, and self-improvement. **Memory is to Agents what parameters are to foundation models** [31, 32, 43, 47, 56]. Recent memory benchmarks have made this bottleneck explicit, evaluating agents on ultra-long conversational consistency, multi-session reasoning, temporal reasoning, knowledge updating, selective forgetting, abstention, and hallucination control [3, 7, 14, 25, 29].

In engineering practice, RAG has become the dominant non-parametric interface for extending language models with external memory, alleviating the static nature of parametric knowledge and the limited size of context windows [22]. Yet standard RAG usually retrieves independent text chunks, whereas long-term agent memory often requires recovering evidence distributed across entities, events, aliases, temporal constraints, and multi-hop dependencies. GraphRAG takes an important step by organizing documents, entities, relations, and summaries as graphs, making cross-document dependencies and reasoning paths more explicit [8, 11]. However, for long-horizon agents, graph structure should not merely serve as an external retrieval index. In this work, we study *agent graph memory* as a coupled write–read–update problem. Given interaction histories or external documents, a memory writer should construct an **evolving graph** whose nodes and edges encode entities, episodes, documents, aliases, temporal constraints, and cross-fragment relations. Given a query, a memory reader should not simply expand from a few matched entities; it should return a **compact, verifiable evidence chain**. The retrieval outcome should further **provide feedback about what the graph lacks**. In other words, the graph is not only built before retrieval and searched afterward; it is the **working substrate through which memory is written, read, corrected, and self-improved**. Around this goal, we identify the following three core challenges.

Challenge I: Agent memory requires global associative reading from fragmented cues. The first challenge is not merely to retrieve text that is semantically similar to the query, but to reconstruct a complete reasoning chain from sparse, fragmented, and sometimes indirect cues. In long-term agent memory, a query may mention only an episodic clue, an alias, or a distant conceptual hint, while the answer depends on intermediate entities that are not explicitly named. Standard vector retrieval tends to return locally similar snippets, and many graph-based retrieval methods start propagation from a small set of query-matched anchor entities. However, if these anchors only cover a local subgraph, the necessary bridge nodes may lie outside the activated region, leaving the evidence chain disconnected even after graph propagation. Thus, agent memory reading should not commit too early to a small set of partial cues [11, 39].

Challenge II: Agent memory requires learned structural use rather than fixed structural expansion. The second challenge is that graph structure should not be used only as a fixed index after graph construction. Many GraphRAG-style systems exploit structure through pre-built communities, paths, graph indexes, or heuristic expansion rules, but once the graph is constructed, the role of structure is largely fixed: a hub remains broadly expanded, a bridge may be missed if it is not reached by the initial anchors, and noisy shortcuts may be treated similarly to useful evidence edges. This is insufficient for agent memory, where the graph itself is continuously updated by new interactions and where the same topological pattern may have different meanings across domains. A structure-aware reader should therefore learn how structural roles affect retrieval [8, 11, 27, 28].

The example in Figure 1 illustrates both challenges. Given the query, the explicit cues are only *Alice*, *lab meeting*, *Cornu Ammonis*, and *agent memory*. A retrieval system that only anchors on the most query-matched nodes may retrieve the meeting note or the biological cue, but still fail to connect them to *hippocampus*,

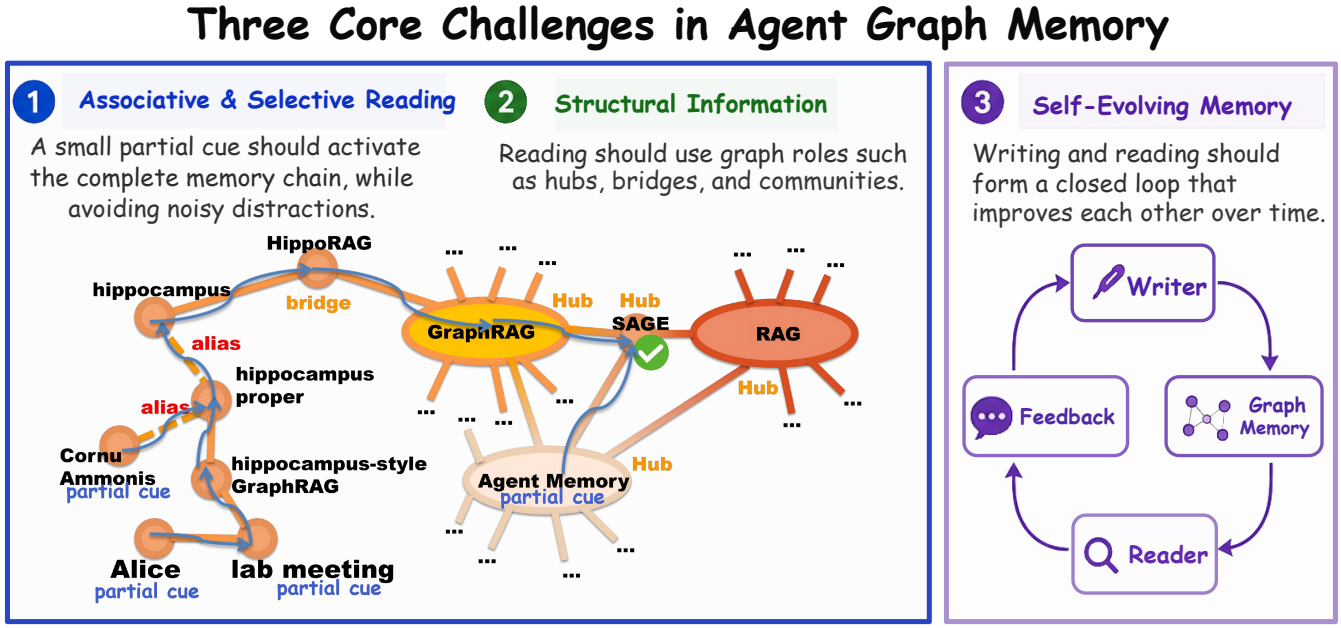

Figure 1: Overview of the three core challenges in agent graph memory, illustrated with a concrete example. Given the query, *"Alice mentioned a work in last week's lab meeting that seemed to be inspired by the Cornu Ammonis. Among works in the same field as that work, are there any that can also help with agent memory? Give one example,"* the memory reader must address associative and selective reading by expanding sparse partial cues into the correct evidence chain while avoiding noisy distractors. It must then exploit structural information in the memory graph, such as aliases, bridges, and hubs, to traverse from Cornu Ammonis to SAGE. Self-evolving memory highlights the closed loop between writing and reading.

HippoRAG, *GraphRAG*, and finally SAGE. This is the associative-reading challenge: the system must piece together a long chain from scattered cues. At the same time, the correct path depends on structural roles: *HippoRAG* is a bridge, *GraphRAG* and *RAG* are hubs that must be controlled rather than blindly expanded, and the edge from *GraphRAG* to SAGE is critical for reaching the final answer. This is the structural-information challenge: the reader must use graph topology in a learned and selective way, not simply propagate uniformly over a fixed graph.

Challenge III: Existing methods mostly optimize retrieval trajectories, but rarely optimize the self-evolution of the memory system itself. Existing RAG and GraphRAG systems often assume that the external memory graph or knowledge base is already available, so the main problem becomes how to retrieve from it. For long-term agents, however, writing is itself part of the memory problem. Conversely, retrieval failures provide useful signals about what the memory graph lacks. For example, if the reader repeatedly needs to traverse from *Cornu Ammonis* to *hippocampus-style GraphRAG* and then to the GraphRAG literature, the memory system should gradually add or strengthen useful structural links, such as a more direct edge from *hippocampus-style GraphRAG* to *GraphRAG*. Thus, a true agent memory system should not only optimize retrieval trajectories; it should optimize the memory graph itself through a closed loop in which better reading exposes writing deficiencies, and better writing makes future reading more accurate, selective, and efficient [3].

To address these challenges, we propose SAGE, a ***S***elf-evolving ***A***gentic ***G***raph-memory ***E***ngine. Unlike GraphRAG systems that mainly use graphs as retrieval middleware, SAGE treats the graph as a dynamic long-term memory object. It couples two mutually reinforcing components: a memory writer incrementally constructs and revises graph memory; a Graph Foundation Model-based memory reader to perform retrieval and provide feedback to the memory writer. SAGE directly targets the three challenges above: it recovers long reasoning chains from fragmented cues, learns how to use structural roles rather than propagate uniformly, and continuously improves the graph memory itself for future queries.

2 Related Work

Retrieval-Augmented Generation and GraphRAG.. RAG provides a non-parametric interface for language models by retrieving external evidence before generation [22]. Many variants further improve retrieval timing, reasoning interaction, adaptive policies, and hierarchical organization [1, 19, 20, 37, 39]. GraphRAG enables structured retrieval over cross-document dependencies and multi-hop evidence paths [8, 9, 11–13, 24, 28, 41, 45, 54, 55]. Another line of work improves retrieval by optimizing retrieval trajectories, including interleaved retrieval and reasoning, self-reflective retrieval, adaptive retrieval, multi-agent RAG, and reinforcement-learning-based query rewriting [1, 2, 5, 19, 39, 40].

Agent Memory. Agent memory studies how LLM-based agents store, update, retrieve, and use past experiences [6, 16, 21, 31, 32, 34, 44, 46, 51–53, 56]. Recent surveys also highlight the importance of human-inspired and graph-based memory mechanisms for LLM

agents [43, 47]. Meanwhile, memory benchmarks evaluate long-term consistency, event reasoning, multi-session reasoning, temporal reasoning, knowledge updating, selective forgetting, abstention, and hallucination control [3, 7, 14, 25, 29].

Graph Foundation Models. Graph Foundation Models (GFMs) aim to learn transferable graph representations through large-scale pre-training, allowing models to reuse structural priors and semantic patterns across graphs, tasks, and domains [27]. Representative early works include GCC, GPT-GNN, and GraphCL, which learn transferable graph representations through cross-network contrast, generative graph pretraining, and graph augmentation based contrastive learning [15, 33, 49, 50].

3 Preliminary

Given a knowledge-intensive memory sample $x = (q, \mathcal{D}, \mathcal{D}^+, y)$, where q denotes the query, $\mathcal{D} = \{d_i\}_{i=1}^N$ denotes the set of candidate historical memory fragments, $\mathcal{D}^+ \subseteq \mathcal{D}$ denotes the gold evidence set that supports the answer, and y denotes the ground-truth answer. The writer is viewed as a structured policy model: at step h, given state s_h, the policy samples a writing action $a_h \sim \pi_\theta(\cdot \mid s_h)$ and updates the partial graph as $\mathcal{G}_{h+1} = \mathcal{G}_h \oplus a_h$. The memory reader $\mathcal{R}_\phi$ performs query-conditioned propagation over the graph, obtains entity relevance scores $\mathbf{s}_E = f_\phi(q, \mathcal{G}) \in \mathbb{R}^{|\mathcal{V}_E|}$, and then projects them into memory-fragment scores. The reader finally outputs $(\widehat{\mathcal{D}}_k, \widehat{\mathcal{G}}_q, \Pi_q) = \mathcal{R}_\phi(q, \mathcal{G}, \mathbf{M})$, where $\widehat{\mathcal{D}}_k = \mathrm{TopK}_{d\in\mathcal{D}}(\mathbf{s}_D(d))$, $\widehat{\mathcal{G}}_q$ is the query-activated subgraph, and Π_q denotes optional relational paths. The generation model then produces the answer $\widehat{y} = \mathrm{LLM}(q, \widehat{\mathcal{D}}_k, \Pi_q)$.

4 Method

At a high level, our method builds a self-evolving graph memory pipeline (Figure 2). The memory writer $\mathcal{W}_\theta$ first transforms the query and candidate historical memory fragments into a heterogeneous graph memory $\mathcal{G}$. The memory reader $\mathcal{R}_\phi$ then performs query-conditioned activation over $\mathcal{G}$: it softly locates query-relevant entities, propagates evidence signals through relational structures, and projects the activated entity-level information back to memory fragments.

4.1 Memory Writer: Graph Memory Writing via Reading Feedback

Policy-based writing. The writer is modeled as a sequential decision-making policy. At step t, the state is defined as $s_t = \left(q, \mathcal{D}, \mathcal{G}_{t-1}, \mathcal{D}_{t-1}^{\mathrm{proc}}\right)$, where $\mathcal{G}_{t-1}$ is the partially written graph, and $\mathcal{D}_{t-1}^{\mathrm{proc}}$ denotes the set of processed documents. The action a_t contains entity-relation triples (u, r, v) together with their source anchors $(u, \mathtt{source}, d)$. Detailed implementation information is provided in Appendix O.

Reader-aware Writing Reward. The writer's reward stems from the task utility of its written graph after being accessed by the memory reader. Given the current graph $\mathcal{G}$, the frozen reader returns the evidence $P_k(q, \mathcal{G})$. Inspired by [40], we employ two complementary types of rewards. The first category measures whether the graph is sufficient as a knowledge carrier to support the derivation of the answer: $r_{\mathrm{ded}}(q, y, \mathcal{G}) = \mathbb{I}\left[\mathrm{Judge}\left(q, y \mid P_k(q, \mathcal{G})\right) = \mathtt{Yes}\right]$. The second category measures whether the graph can serve as a knowledge index to recover the supporting text: $r_{\mathrm{rec}}(q, \mathcal{D}^+, \mathcal{G}) = \frac{|P_k(q,\mathcal{G}) \cap \mathcal{D}^+|}{|\mathcal{D}^+|}$, $r_{\mathrm{pre}}(q, \mathcal{D}^+, \mathcal{G}) = \frac{|P_k(q,\mathcal{G}) \cap \mathcal{D}^+|}{|P_k(q,\mathcal{G})|}$. Where r_{rec} encourages the coverage of necessary evidence, while r_{pre} penalizes the expansion of irrelevant evidence. To align with end-to-end question answering, we also use an answer-level auxiliary reward $r_{\mathrm{ans}}(q, y, \mathcal{G}) = \max_{y' \in \mathcal{Y}(y)} \mathrm{F1}(\hat{y}, y'), \hat{y} = \mathrm{LLM}\left(q, P_k(q, \mathcal{G})\right)$, where $\mathcal{Y}(y)$ is the set of answer aliases. In practice, we adopt a hybrid task reward $r_{\mathrm{task}} = \frac{\alpha r_{\mathrm{rec}} + \beta r_{\mathrm{pre}} + \gamma r_{\mathrm{ded}}}{\alpha+\beta+\gamma}$.

Furthermore, to prevent the policy from inflating the graph size by stacking duplicate triples, we define a repetition rate: $\rho_{\mathrm{rep}}(\mathcal{G}) = \frac{|\mathcal{T}(\mathcal{G})| - |\,\mathrm{uniq}(\mathcal{T}(\mathcal{G}))|}{|\mathcal{T}(\mathcal{G})|}$ and derive the trajectory return $R(\tau) = r_{\mathrm{task}}(\tau) - \lambda_{\mathrm{rep}}\rho_{\mathrm{rep}}(\mathcal{G}_\tau) + \lambda_{\mathrm{fmt}} \sum_{t=1}^{|\tau|} r_t^{\mathrm{fmt}}$. This directly addresses the issue revealed by works such as HaluMem: errors in memory systems often do not emerge only at the answering stage, but are already written during the extraction and updating phases [3]. We employ standard clipped GRPO to update the writer.

4.2 Memory Reader: Memory Retrieval Based on Graph Foundation Model

The memory reader must operate stably over graph memory that is continuously updated by the writer. Dense retrievers mainly learn query–document semantic matching and thus struggle to exploit entity roles, bridge paths, and cross-community dependencies, while conventional GNN retrievers are often tied to fixed graph distributions and generalize poorly across domains, users, and evolution stages. We therefore adopt a Graph Foundation Model (GFM) as the memory reader, whose multi-graph pre-training enables transferable structural priors and lightweight calibration on new graphs [28, 54]. Formally, the memory reader outputs an entity distribution, a document distribution, and an optional retrieval subgraph $f_\phi(q, \mathcal{G}, \mathcal{D}) = \left(p_\phi(e \mid q, \mathcal{G}), p_\phi(d \mid q, \mathcal{G}, \mathcal{D}), \mathcal{G}_q\right)$. Where $p_\phi(e \mid q, \mathcal{G})$ represents the entity memory activated by the query, $p_\phi(d \mid q, \mathcal{G}, \mathcal{D})$ denotes the final retrieved textual evidence, and $\mathcal{G}_q$ provides an interpretable retrieval path. To obtain a compact and query-aligned activated subgraph, we further introduce a lightweight query-conditioned subgraph selector; implementation details are provided in Appendix I.

Cognition-inspired Structured Query Planning. When humans extract long-term memories, the brain often automatically generates multi-dimensional retrieval cues to anchor the target based on only a vague final intention. Inspired by this, we no longer treat the natural language query as a single retrieval command. Instead, we introduce a planning function $\mathcal{P}_\omega$ to simulate the cue reconstruction process of the human brain before awakening memory, decomposing the initial query into a set of rich associative probes: $\mathcal{P}_\omega(q) = \left(\mathcal{E}_{\mathrm{exp}}, \mathcal{A}, C_{\mathrm{rel}}, C_{\mathrm{hard}}, \tau, \{(\tilde{q}_m, \alpha_m, t_m)\}_{m=1}^M\right)$. Detailed definitions of the notation, additional information, and the concrete prompt templates and output schema are provided in Appendix K. This multi-path concurrent awakening method effectively overcomes the "tip-of-the-tongue phenomenon" (i.e., difficulties in alias alignment or missing bridging entities) and naturally stitches together forgotten implicit relationships [1, 39, 44, 53].

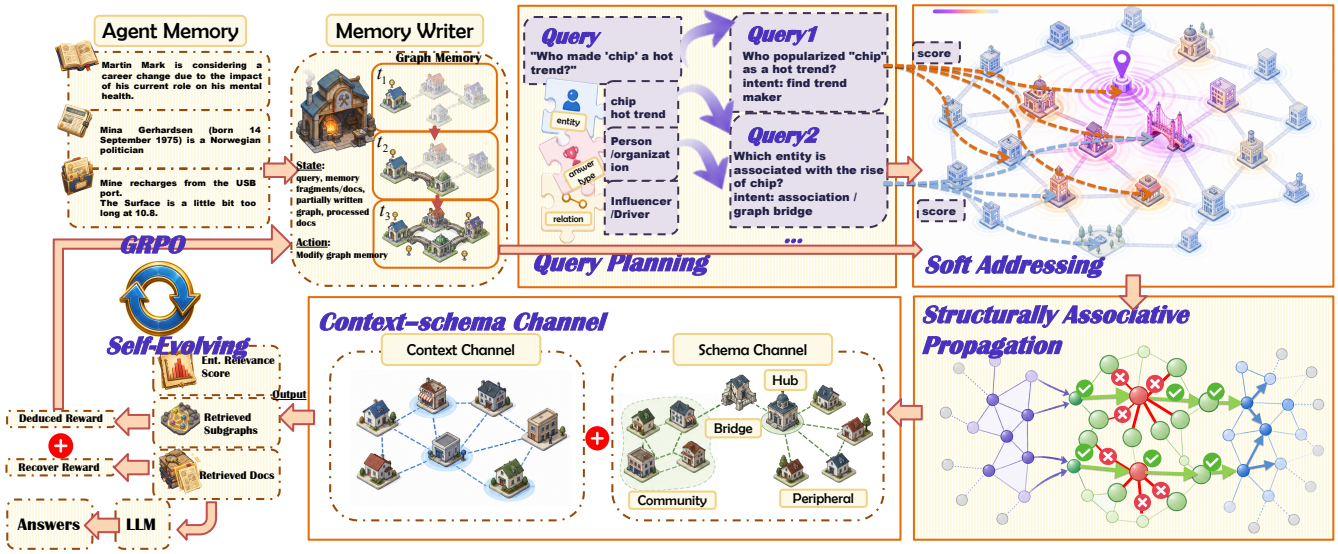

Figure 2: Overall pipeline of the proposed SAGE. The memory writer incrementally constructs and updates graph memory from observations through state-conditioned writing actions, and receives rewards from downstream memory use. The resulting retrieval feedback closes the loop between writing and reading, enabling graph memory to improve over time.

Soft Addressing and Pre-activation of Memory Fragments. Cognitive neuroscience reveals that human memory retrieval involves not only the extraction of perfectly matching information but also the instinctive awakening of peripherally related memories through *Semantic Priming*. And to address the first challenge, we treat the calculation of the query-conditioned entry score $s_e(q)$ as a comprehensive assessment of the stimulus intensity across different *Memory Engrams*:

$$\begin{aligned} s_e(q) =& \lambda_1 \ \mathrm{Exact}(e, \mathcal{E}_{\mathrm{exp}}) + \lambda_2 \ \mathrm{Alias}(e, \mathcal{A}) \\ &+ \lambda_3 \max_{m \le M} \cos\big(\mathrm{Emb}(\mathrm{desc}(e)), \mathrm{Emb}(\tilde{q}_m)\big) \\ &+ \lambda_4 \ \mathrm{Type}(e, \tau) + \lambda_5 \ \mathrm{Cons}(e, C_{\mathrm{hard}}) + \lambda_6 \sum_{\xi \in \mathrm{NER}(q)} \mathrm{EL}(e \mid \xi). \end{aligned} \tag{1}$$

(2)

Subsequently, the system employs a Softmax function with a temperature coefficient T_0 to simulate the brain's limited Attention Allocation mechanism during retrieval. This normalizes the multi-dimensional stimulus signals to form the initial activation distribution of the memory atlas $p_0(e \mid q) = \frac{\exp(s_e(q)/T_0)}{\sum_{v \in \mathcal{V}_E} \exp(s_v(q)/T_0)}$. Based on this distribution, we define the initial state of the memory nodes as $\mathbf{h}_e^{(0)} = \big(p_0(e \mid q)\big)^{\eta} W_q \, \mathrm{Emb}(q) + W_x \mathbf{x}_e$. In this process, $\mathbf{x}_e$ acts as the solidified **long-term memory** (static representation of entities) in the brain, while the query vector adjusted by the cognitive recall degree $p_0(e \mid q)$ represents the current **working memory** (task context).

Synapse-inspired Structurally Conditioned Associative Propagation. To address the second challenge while avoiding indiscriminate diffusion, we introduce edge-level vector structural gating in the GFM. The node-level structural features, edge-pair structural features, and graph-level summary are defined as:

$$\phi(v) = \big[\log(1 + d_v), c_v, \kappa_v, \bar{d}_{\mathcal{N}(v)}\big], \tag{3}$$

$$\psi(u, v) = \big[|d_u - d_v|, |\mathcal{N}(u) \cap \mathcal{N}(v)|, \mathrm{Jaccard}(\mathcal{N}(u), \mathcal{N}(v))\big], \tag{4}$$

$$\mathbf{r}_{\mathcal{G}} = \big[\mathrm{mean}_{v \in \mathcal{V}_E} \phi(v); \mathrm{std}_{v \in \mathcal{V}_E} \phi(v); \mathrm{dens}(\mathcal{G})\big]. \tag{5}$$

Detailed definitions and normalization procedures are provided in Appendix L. The edge structural context for the l-th layer is $\mathbf{z}_{uv}^{(l)} = \big[E_n^{(l)}(\phi(u)); E_n^{(l)}(\phi(v)); E_p^{(l)}(\psi(u,v)); E_g^{(l)}(\mathbf{r}_{\mathcal{G}})\big]$, which generates the vector gating $\mathbf{g}_{uv}^{(l)} = \mathbf{1} + \delta \tanh\big(\mathrm{MLP}_g^{(l)}(\mathbf{z}_{uv}^{(l)})\big)$. Let η_{uv} be the normalized adjacency weight with self-loops; the message and node updates are $\mathbf{m}_{u \to v}^{(l)} = \eta_{uv} \, \mathbf{g}_{uv}^{(l)} \odot W_m^{(l)} \mathbf{h}_u^{(l-1)}$, $\mathbf{h}_v^{(l)} = \mathrm{LayerNorm}\Big(\mathbf{h}_v^{(l-1)} + \mathrm{PReLU}\Big(\mathbf{b}^{(l)} + \sum_{u \in \mathcal{N}(v)} \mathbf{m}_{u \to v}^{(l)}\Big)\Big)$. Unlike traditional heuristic path expansion, PPR walks, or community summarization [8, 9, 41], the system here can actively perform **Inhibition** of non-specific generalized memories (suppressing hub edges), keenly capture and preserve **long-distance associations** across different cognitive clusters (lateral thinking/bridge edge preservation), and undergo **Habituation** (weakening redundant edges) toward highly repetitive local information, much like the human brain.

Traditional query-dependent GNNs or PPR-style expansion can perform multi-hop propagation along graph structures. But the key issue is not simply to expand the propagation range, but to preserve the advantage of query-relevant evidence signal over distractor noise under a limited top-k budget. Proposition 1(i) summarizes this signal–budget view: soft addressing improves the initial evidence activation, structural gating preserves bridge/evidence paths while suppressing noisy neighborhoods, and controlled entity-to-document projection converts the entity-level advantage into more

efficient document-level retrieval. Complete definitions, assumptions, and proofs are provided in Appendix B.

Target Graph Calibration and Cross-graph Structural Priors. Human memory, on one hand, reorganizes cues based on the current context, while on the other, it retains relatively stable structured recall habits. In our self-evolving graph memory, each $\mathcal{G}$ generated by the writer per round alters the local topology and noise distribution; therefore, the reader cannot rely solely on propagation patterns from a fixed graph.

Since the writer continuously changes the memory graph, the reader must simultaneously adapt to the current target graph and preserve cross-graph structural priors. This is precisely why we introduce the context–schema decomposition. As summarized in Proposition 1(ii), the schema channel provides a transferable structural prior, while the context channel corrects the target-graph residual induced by the current writer, current domain, entity granularity, and local noise. The complete theoretical motivation is provided in Appendix C and Appendix D.

First, a feature prompt vector $\mathbf{p}_f$ is used for a lightweight calibration of the query-activated input $\tilde{\mathbf{h}}_e^{(0)} = \mathbf{p}_f \odot \mathbf{h}_e^{(0)}$. The contextual calibration channel performs gated propagation on the current graph $\mathcal{G}$: $\mathbf{H}_{\text{ctx}} = F_{\text{gate}}(\tilde{\mathbf{H}}^{(0)}, \mathcal{G}; \Theta_{\text{gate}})$. Where $\mathbf{H}_{\text{ctx}}$ captures the immediate structural state within the current memory graph. Simultaneously, the schema prior channel maintains a set of cross-graph structural prompt bases $\{\mathbf{P}_j^{(l)}\}_{j=1}^{K}$, which are used to encode stable reading habits formed during multi-graph training: $\omega_j^{(l)} = \text{softmax}_j(\mathbf{a}^{(l)}/T_p), \mathbf{P}_{\text{schema}}^{(l)} = \sum_{j=1}^{K} \omega_j^{(l)} \mathbf{P}_j^{(l)}$. Propagation is executed based on these schema prompts to obtain: $\mathbf{H}_{\text{sch}} = F_{\text{prompt}}\left(\tilde{\mathbf{H}}^{(0)}, \mathcal{G}; \{\mathbf{P}_{\text{schema}}^{(l)}\}_{l=1}^{L}\right)$. The final entity representation is jointly determined by the current context and the long-term schema: $\mathbf{H}(q, \mathcal{G}) = \mathbf{H}_{\text{ctx}} + \beta_{\text{sch}} \mathbf{H}_{\text{sch}}$. Here, $\mathbf{H}_{\text{ctx}}$ is analogous to a context-dependent immediate recall state, responsible for adapting to the specific graph structure generated by the current writer; $\mathbf{H}_{\text{sch}}$ is akin to a memory schema formed across experiences, retaining the ability to recognize stable patterns such as bridge nodes, community boundaries, core–periphery structures, and noise short-circuits.

Reader Training. Reader training aims to learn cross-graph transferable retrieval biases through a two-stage procedure. First, we perform structural contrastive pre-training on multiple augmented graph views. Then, in the supervised fine-tuning stage, we align these transferable capabilities with question-driven evidence retrieval by training the reader to identify and rank supporting entities for each query using weighted classification and multi-positive ranking objectives. Implementation details are provided in Appendices M and N.

Writer–Reader Self-evolution. To address the third challenge, we propose a self-evolution framework. Each of our self-evolution iterations consists of two phases. First, we fix the reader and train the writer using its retrieval results as rewards. Subsequently, we use the updated writer to generate new graphs and continue training the reader. The overall procedure is detailed in Algorithm 1.

Table 1: Open-domain retrieval results on NQ and PopQA. We report passage/document-level Recall (%) at top-2 and top-5 when comparable numbers are available in original papers or later works that reproduce/cite these methods. Best available results are in bold and runner-ups are underlined. Only rows marked with 0-shot are our zero-shot transfer results; baseline rows are not marked as zero-shot.

Zero-shot setting applies only to SAGE on NQ and PopQA.

Dataset	NQ		PopQA	
Method	$R@2_D$	$R@5_D$	$R@2_D$	$R@5_D$
BM25 (▷ *SIGIR'94*)	28.2†	56.1†	24.0†	35.7†
Contriever (▷ *TMLR'22*)	29.1†	54.6†	27.0†	43.2†
GTR (▷ *EMNLP'22*)	35.0†	63.4†	40.1†	49.4†
ColBERTv2 (▷ *NAACL'22*)	36.8⋆	64.3⋆	–	–
RAPTOR (▷ *ICLR'24*)	40.3†	68.3†	40.2†	48.7†
Proposition (▷ *EMNLP'24*)	33.1⋆	62.2⋆	–	–
HippoRAG (▷ *NeurIPS'24*)	21.3†	44.4†	40.0†	<u>53.8</u>†
HippoRAG 2 (▷ *ICML'25*)	<u>45.6</u>†	<u>78.0</u>†	**43.9**†	51.7†
PropRAG (▷ *EMNLP'25*)	–	77.9‡	–	**56.2**‡
SAGE (ours) 0-shot	**82.5** 0-shot	**91.6** 0-shot	<u>41.5</u> 0-shot	52.3 0-shot

† Values are from the reproduced passage Recall@2/5 evaluation in ▷. ‡ Values are from the Recall@5 table in ▷; Recall@2 is not reported there. ⋆ Values are from the reproduced single-step retrieval table in ▷; PopQA is not reported there.

Proposition 1. Theoretical consequences of SAGE

(i) **Signal–budget efficiency.** Soft addressing, structural gating, and controlled entity-to-document projection jointly improve evidence signal over distractor noise, thereby reducing the top-k budget needed for evidence coverage.
[▷ SNR proof] [▷ Budget proof]

(ii) **Context–schema decomposition.** The reader combines transferable structural priors with target-graph calibration, so adaptation only needs to correct graph-specific residuals.
[▷ Proof]

(iii) **Evolution stability.** Under bounded graph drift, consecutive writer updates induce bounded document-score changes.
[▷ Proof]

From a theoretical perspective, this process can be interpreted as approximate coordinate improvement over a joint memory utility: the writer update improves the readability of the graph memory, while the reader update reduces writer-induced graph distribution shift and reward bias.

We provide the full coordinate-improvement result, the surrogate reward bias bound, and the analysis of single-sided update bottlenecks in Appendix F. In addition, Proposition 1(iii) shows that although each writer update changes the graph structure in self-evolving memory, the reader output does not oscillate arbitrarily with graph evolution. We provide detailed training, inference, memory, and selector-regularizer complexity analyses in Appendix J.

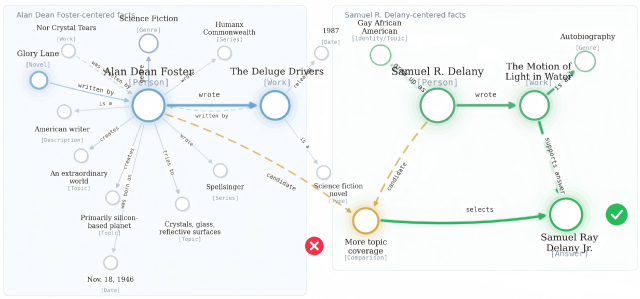

Figure 3: Visualization of the retrieved results.

5 Experiments

This section presents an experimental evaluation centered around four research questions (RQs). ***RQ1***: whether SAGE can bring consistent benefits in tasks such as multi-hop QA and open-domain transfer;

RQ2: whether SAGE is an agent memory system capable of handling long-term conversation history, knowledge updates, and memory hallucination;

RQ3: whether the writer–reader closed loop truly yields self-evolution benefits;

RQ4: further analysis of where and how the performance gains come from specific designs.

Datasets. We evaluate SAGE on five complementary scenarios. The first category consists of general QA benchmarks and three multi-hop QA benchmarks, including NQ, PopQA, HotpotQA, 2WikiMultiHopQA, and MuSiQue, used to examine whether the system can recover bridge entities across documents and combine evidence and reasoning paths. The second category focuses on a practical e-commerce application scenario, using a Review-Based Question Answering Task: AmazonQA, to assess its value in real e-commerce applications with real noisy reviews. The third category comprises long-term agent memory datasets, including LongMemEval and HaluMem, used to test information extraction from long interaction histories, multi-session reasoning, temporal reasoning, knowledge updating, abstention, and operation-level hallucination. Table 13 summarizes the details of each dataset. Further details on baselines and metrics can be found in Appendix R.

5.1 End-to-End Effectiveness

Multi-hop Question Answering. Table 3 reports the main results on general QA benchmarks and three multi-hop QA benchmarks. Table 9 reports the results of retrieval performance on multi-hop QA benchmarks. It is worth mentioning that even when we directly test on NQ and PopQA using a model trained only on MuSiQue, HotpotQA, and 2WikiMultiHopQA, we still achieve very strong performance, especially on NQ; see Table 1 for the detailed results.

Domain-specific Memory. Table 10 reports the results on AmazonQA. SAGE consistently outperforms the neural baseline R-Net across all metrics, indicating strong cross-task generalization. After training on AmazonQA, Ours achieves substantial gains. Overall, training and interaction rounds steadily enhance performance, while the zero-shot results demonstrate promising transfer ability.

[1] `memobase`: https://github.com/memodb-io/memobase.
[2] `Supermemory`: https://github.com/supermemoryai/supermemory.
[3] `MemU`: https://github.com/NevaMind-AI/memU.

Table 2: Performance of representative memory systems on `LongMemEval`. We report accuracy (%) on six task categories: single-session user (SS-U), single-session assistant (SS-A), merged single-session recall (SSR), single-session preference (SS-P), knowledge update (KU), temporal reasoning (TR), and multi-session reasoning (MS). SSR is computed as the weighted average of SS-U and SS-A when both are available; if a source reports only merged single-session recall, SS-U/SS-A are left blank. Best results are in bold and runner-ups are underlined. The darker the cell, the better. Results are grouped by reporting protocol and should not be treated as a single strict leaderboard. Only rows marked with [0-shot] are our zero-shot transfer results; baseline rows and trained variants are not marked as zero-shot.

Zero-shot setting applies only to `Ours` rows marked with [0-shot] on `LongMemEval`.

Dataset	LongMemEval-S / LongMemEval							
Method	SS-U	SS-A	SSR	SS-P	KU	TR	MS	Overall
Unified protocol in TiMem (GPT-4o-mini, LLJ accuracy)								
MemoryBank (▷)	50.0	9.8	32.1	0.0	21.8	0.0	0.0	11.5
A-MEM (▷)	82.9	**87.5**	84.9	39.3	72.8	36.1	40.3	55.4
Mem0 (▷)	94.3	51.8	75.4	50.0	78.7	49.2	66.2	65.0
MemoryOS (▷)	81.1	78.2	79.8	51.3	56.1	53.4	44.8	58.1
MemOS (▷)	93.7	67.9	82.2	50.7	76.7	65.1	58.8	68.7
TiMem (▷)	**95.7**	82.1	**89.7**	63.3	86.2	68.4	**70.8**	**76.9**
MemOS evaluation suite (short-answer prompt)								
MIRIX (▷)	72.8	63.6	68.8	53.3	52.6	25.6	30.1	43.5
Mem0 (▷)	82.9	26.8	57.9	**90.0**	66.7	72.2	63.1	66.4
Zep (▷)	92.9	75.0	84.9	53.3	74.4	54.1	47.4	63.2
memobase[1]	92.8	23.2	61.9	80.0	**89.7**	**75.9**	66.9	72.4
Supermemory[2]	85.7	58.9	73.8	**90.0**	55.1	44.4	52.6	58.4
MemU[3]	67.1	19.6	46.0	76.7	41.0	17.3	42.1	38.4
Our method								
Ours (0-shot)[0-shot]	60.3	81.4	68.0	16.7	23.5	13.8	9.4	28.4
Ours (trained)	73.3	80.0	76.0	23.1	27.8	22.8	12.5	34.3
Ours +1 round	79.4	80.5	79.6	15.8	26.2	22.8	10.7	34.1

5.2 Long-term Agent Memory Evaluation

The LongMemEval results are shown in Table 2. The HaluMem results are shown in Table 11. SAGE is compared against highly specialized long-term memory systems, making this a challenging evaluation setting. Although SAGE does not yet surpass the strongest system-level baselines. Notably, SAGE +1 round already outperforms Memobase on several metrics, suggesting that it is competitive despite being less system-engineered. The remaining gap mainly lies in memory updating and high-coverage extraction, indicating clear potential for further gains with stronger memory management and update mechanisms.

5.3 Further Analysis

As shown in Table 4, SAGE demonstrates a strong speed advantage. It achieves the fastest retrieval time, indicating strong potential for practical and large-scale deployment. To further analyze the interpretability of SAGE, we visualize the retrieved subgraph for a representative case, as shown in Figure. A detailed case study can be found in P.1. The detailed ablation study design, analysis,

Table 3: Results of multi-hop question answering (QA) performance. We report Exact Match (EM) and F1 score, both reported as percentages (%). Best results are in bold and runner-ups are underlined. The darker the cell, the better.

Dataset	HotpotQA		MuSiQue		2WikiMultiHopQA		Avg. Rank
Method	EM	F1	EM	F1	EM	F1	
BM25 (▷ *arXiv'24*)	40.0	53.2	19.5	23.6	46.9	57.9	15.5
Contriever (▷ *TMLR'22*)	34.9	51.2	16.3	21.6	24.3	33.9	20.5
GTR (▷ *EMNLP'22*)	33.8	51.9	15.1	25.2	33.7	42.5	19.7
ColBERTv2 (▷ *NAACL'22*)	43.4	57.7	15.5	26.4	33.4	43.3	17.3
RAPTOR (▷ *ICLR'24*)	48.2	59.2	17.6	28.9	30.6	42.0	15.8
GraphRAG (▷ *arXiv'24*)	35.3	54.6	13.4	29.5	28.3	46.9	18.0
G-Retriever (▷ *NeurIPS'24*)	33.2	50.3	18.0	25.9	42.3	45.6	18.2
LightRAG (▷ *arXiv'24*)	36.8	48.3	18.1	27.5	45.1	49.5	16.7
HippoRAG (▷ *NeurIPS'24*)	41.8	55.0	19.2	29.8	46.6	59.5	13.8
HippoRAG 2 (▷ *ICML'25*)	57.3	69.6	33.9	40.5	**75.4**	78.2	4.7
SubgraphRAG (▷ *ICLR'25*)	52.6	65.8	33.0	39.2	71.0	76.2	7.5
PropRAG (▷ *EMNLP'25*)	57.2	70.1	**38.9**	41.3	73.0	79.9	3.8
GFM-RAG (▷ *NeurIPS'25*)	51.6	66.9	30.2	40.4	69.8	77.7	7.8
FLARE (▷ *EMNLP'23*)	48.7	60.6	16.2	28.4	46.7	65.4	12.7
Adaptive-RAG (▷ *NAACL'24*)	45.5	59.6	13.8	25.6	48.9	62.8	14.3
ColBERTv2 + IRCoT (▷ *ACL'23*)	45.5	58.4	19.1	30.5	35.4	45.1	14.4
HippoRAG + IRCoT (▷ *ACL'23*)	45.7	59.2	21.9	33.3	47.7	62.7	11.3
GFM-RAG + IRCoT (▷ *ACL'23*)	56.0	71.8	36.6	49.2	72.5	80.8	3.7
SAGE (ours)	51.3	72.7	28.0	47.3	64.5	75.3	7.5
ours +1 round	**60.4**	73.4	32.2	52.3	71.7	79.5	4.2
ours +2 round	56.1	**80.8**	34.2	**53.1**	74.0	80.4	**2.5**
ours + IRCoT	52.3	75.0	35.2	52.6	72.2	**82.9**	3.3

Table 4: Retrieval efficiency comparison. We report retrieval time in seconds on HotpotQA, MuSiQue, and 2Wiki. For Time, lower is better. Best results are in bold and runner-ups are underlined. The darker the cell, the better.

Dataset	HotpotQA	MuSiQue	2Wiki
Method	Time↓	Time↓	Time↓
Single-step retrieval methods			
ColBERTv2	0.035	**0.030**	0.029
HippoRAG	0.255	0.251	0.158
LightRAG	0.861	1.109	0.911
GraphRAG (MS)	2.759	3.037	1.204
GFM-RAG	0.107	0.124	0.060
Iterative retrieval methods			
IRCoT + ColBERTv2	1.146	1.152	2.095
IRCoT + HippoRAG	3.162	3.104	3.441
SAGE	**0.032**	0.034	**0.019**

and results for the Memory Writer and Reader can be found in Appendix H and Appendix G, respectively.

6 Conclusion

We presented SAGE, a self-evolving agentic graph-memory engine that treats memory as a dynamic substrate for writing, reading, and continual improvement. Experiments show that SAGE improves evidence recovery, grounding, and retrieval efficiency, suggesting that self-evolving graph memory is a promising foundation for long-horizon language agents.

References

[1] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

[2] Sungguk Cha, DongWook Kim, Taeseung Hahn, Mintae Kim, Youngsub Han, and Byoung-Ki Jeon. 2025. Annotation-Free Reinforcement Learning Query Rewriting via Verifiable Search Reward. *arXiv preprint arXiv:2507.23242* (2025).

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

[4] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense x retrieval: What retrieval granularity should we use?. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 15159–15177.

[5] Yiqun Chen, Lingyong Yan, Weiwei Sun, Xinyu Ma, Yi Zhang, Shuaiqiang Wang, Dawei Yin, Yiming Yang, and Jiaxin Mao. 2025. Improving retrieval-augmented generation through multi-agent reinforcement learning. *arXiv preprint arXiv:2501.15228* (2025).

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

[7] Hongwei Wang Di Wu, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu Longmemeval. [n. d.]. Benchmarking chat assistants on long-term interactive memory, 2024. *URL https://arxiv. org/abs/2410.10813* 2 ([n. d.]), 14.

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

[9] Zirui Guo, Lianghao Xia, Yanhua Yu, Tian Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779* 2, 3 (2024).

[10] Mansi Gupta, Nitish Kulkarni, Raghuveer Chanda, Anirudha Rayasam, and Zachary C Lipton. 2019. Amazonqa: A review-based question answering task. *arXiv preprint arXiv:1908.04364* (2019).

[11] Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems* 37 (2024), 59532–59569.

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

[13] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems* 37 (2024), 132876–132907.

[14] Yuanzhe Hu, Yu Wang, and Julian McAuley. 2025. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257* (2025).

[15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1857–1867.

[16] Zhengjun Huang, Zhoujin Tian, Qintian Guo, Fangyuan Zhang, Yingli Zhou, Di Jiang, Zeying Xie, and Xiaofang Zhou. 2025. Licomemory: Lightweight and cognitive agentic memory for efficient long-term reasoning. *arXiv preprint arXiv:2511.01448* (2025).

[17] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. GPT-4o system card. *arXiv preprint arXiv:2410.21276* (2024).

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

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

[20] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*. 7969–7992.

[21] Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 25972–25981.

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[23] Kai Li, Xuanqing Yu, Ziyi Ni, Yi Zeng, Yao Xu, Zheqing Zhang, Xin Li, Jitao Sang, Xiaogang Duan, Xuelei Wang, et al. 2026. TiMem: Temporal-Hierarchical Memory Consolidation for Long-Horizon Conversational Agents. *arXiv preprint arXiv:2601.02845* (2026).

[24] Mufei Li, Siqi Miao, and Pan Li. 2024. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. *arXiv preprint arXiv:2410.20724* (2024).

[25] Yifei Li, Weidong Guo, Lingling Zhang, Rongman Xu, Muye Huang, Hui Liu, Lijiao Xu, Yu Xu, and Jun Liu. 2026. Locomo-Plus: Beyond-Factual Cognitive Memory Evaluation Framework for LLM Agents. *arXiv preprint arXiv:2602.10715* (2026).

[26] Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, et al. 2025. Memos: An operating system for memory-augmented generation (mag) in large language models. *arXiv preprint arXiv:2505.22101* (2025).

[27] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. 2025. Graph foundation models: Concepts, opportunities and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2025).

[28] Linhao Luo, Zicheng Zhao, Gholamreza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. 2025. GFM-RAG: graph foundation model for retrieval augmented generation. *arXiv preprint arXiv:2502.01113* (2025).

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

[30] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, et al. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 9844–9855.

[31] Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. MemGPT: towards LLMs as operating systems. (2023).

[32] Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*. 1–22.

[33] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1150–1160.

[34] 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* (2025).

[35] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*. Springer, 232–241.

[36] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 3715–3734.

[37] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

[38] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554.

[39] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*. 10014–10037.

[40] Hong Ting Tsang, Jiaxin Bai, Haoyu Huang, Qiao Xiao, Tianshi Zheng, Baixuan Xu, Shujie Liu, and Yangqiu Song. 2025. Autograph-r1: End-to-end reinforcement learning for knowledge graph construction. *arXiv preprint arXiv:2510.15339* (2025).

[41] Jingjin Wang and Jiawei Han. 2025. Proprag: Guiding retrieval with beam search over proposition paths. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. 6223–6238.

[42] Yu Wang and Xi Chen. 2025. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957* (2025).

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

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

[45] Tianyang Xu, Haojie Zheng, Chengze Li, Haoxiang Chen, Yixin Liu, Ruoxi Chen, and Lichao Sun. 2025. NodeRAG: Structuring graph-based rag with heterogeneous nodes. *arXiv preprint arXiv:2504.11544* (2025).

[46] 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* (2025).

[47] Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Zhishang Xiang, et al. 2026. Graph-based Agent Memory: Taxonomy, Techniques, and Applications. *arXiv preprint arXiv:2602.05665* (2026).

[48] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William 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*. 2369–2380.

[49] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

[50] Xingtong Yu, Zechuan Gong, Chang Zhou, Yuan Fang, and Hui Zhang. 2025. Samgpt: Text-free graph foundation model for multi-domain pre-training and cross-domain adaptation. In *Proceedings of the ACM on Web Conference 2025*. 1142–1153.

[51] Juwei Yue, Chuanrui Hu, Jiawei Sheng, Zuyi Zhou, Wenyuan Zhang, Tingwen Liu, Li Guo, and Yafeng Deng. 2026. HyperMem: Hypergraph Memory for Long-Term Conversations. *arXiv preprint arXiv:2604.08256* (2026).

[52] Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025. G-memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398* (2025).

[53] Kai Zhang, Xinyuan Zhang, Ejaz Ahmed, Hongda Jiang, Caleb Kumar, Kai Sun, Zhaojiang Lin, Sanat Sharma, Shereen Oraby, Aaron Colak, et al. 2025. Assomem: Scalable memory qa with multi-signal associative retrieval. *arXiv preprint arXiv:2510.10397* (2025).

[54] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, et al. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958* (2025).

[55] Yibo Zhao, Jiapeng Zhu, Ye Guo, Kangkang He, and Xiang Li. 2025. Eˆ 2GraphRAG: Streamlining Graph-based RAG for High Efficiency and Effectiveness. *arXiv preprint arXiv:2505.24226* (2025).

[56] Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 38. 19724–19731.

Algorithm 1: Writer–Reader Self-evolution Training for SAGE

Input: Training set $\mathcal{D}_{\text{train}}$, writer π_{θ_0}, GFM reader f_{ϕ_0}, self-evolution iterations T
Output: Trained writer π_{θ_T} and reader f_{ϕ_T}

for $t = 0, \ldots, T-1$ **do**
  // Writer update: fixed GFM reader as reward environment
  for *each sample* $x = (q, \mathcal{D}, \mathcal{D}^+, y) \in \mathcal{D}_{\text{train}}$ **do**
    Sample G graph construction trajectories $\{\tau_i\}_{i=1}^{G}$ from π_{θ_t};
    for $i = 1, \ldots, G$ **do**
      Obtain graph $\mathcal{G}_i$ and retrieve $P_k(q, \mathcal{G}_i)$ using f_{ϕ_t};
      Calculate return R_i;
    Update writer π_{θ_t};
  // Reader update: improved graphs as memory substrate
  Construct a set of graph memories $\{\mathcal{G}_x\}$ for the training corpus using $\pi_{\theta_{t+1}}$;
  Update GFM reader f_{ϕ_t} on $\{\mathcal{G}_x\}$;

A Additional Analysis of the Memory Writer

This appendix provides a detailed analysis of the memory writer experiments in the main text.

A.1 Reward Design and Writer Behavior

In Table 5, different RL rewards induce different writer behaviors. GFM-pretrained-only achieves Precision/Recall/Deducible of 0.838/0.818/0.510, while GFM-finetuned achieves 0.824/0.813/0.512, indicating that relying solely on supervised finetuning cannot stably improve the utility of graph memory for a frozen reader. This result is also consistent with our setup: the goal of the memory writer is not to reproduce a static graph format.

RL-Recall improves Precision and Recall to 0.889/0.835, but Deducible drops to 0.502. This shows that rewarding only supporting context coverage encourages the writer to store more locally relevant evidence, but does not necessarily lead to a complete multi-hop reasoning chain. RL-F1 further raises Recall to 0.881, but Deducible is only 0.497, again indicating a gap between retrieval matching quality and answer deducibility: the reader hitting the supporting contexts does not guarantee that these contexts are organized in a way sufficient to support answer reasoning. In contrast, RL-Deduce achieves 0.861/0.892/0.517, showing that using answer deducibility directly as feedback can encourage the writer to focus more on bridging entities, cross-document relations, and answer-relevant causal or attribute paths.

RL-Hybrid achieves Precision and Recall of **0.902** and **0.917**, respectively, representing improvements of +0.064 and +0.099 over pretrained-only, while Deducible reaches 0.522. This indicates that hybrid rewards can mitigate the bias of a single reward: they both avoid the introduction of too much weakly relevant evidence caused by a pure recall reward and prevent a pure deducibility reward from overfavoring short paths or local answer clues. Hybrid + frozen answer API achieves the highest Deducible, at **0.526**, but Precision and Recall drop to 0.832/0.874. This suggests that stronger answer-side feedback can further improve reasoning usability, but it may also make the writer more conservative, writing only evidence directly related to the final answer and thereby sacrificing some supporting context coverage.

A.2 Cross-domain Transfer

Table 6 shows that the base writer trained on HotpotQA/MuSiQue has a certain degree of transferability to new domains, but training on the target domain remains very important. On GRBench, Base→GRBench achieves Precision/Recall/Deducible of 0.575/0.609/0.411, while GRBench train→val improves to 0.794/0.833/0.596. This improvement indicates that the writing strategy learned for multi-hop QA can transfer to structured product or domain graph memory tasks, but the entity types, attribute relations, and evidence granularity in the target domain still need to be re-adapted.

On HaluMem and LongMemEval, cross-domain differences are even more pronounced. The Base→HaluMem results are 0.230/0.448/0.299, which improve to 0.312/0.708/0.438 after training on the target domain; the Base→LongMemEval results are 0.232/0.376/0.475, which improve to 0.377/0.439/0.531 after training on the target domain. These results indicate that memory writing in agent memory tasks requires not only extracting explicit facts, but also maintaining user preferences, temporal order, state updates, and long-term consistency. In traditional multi-hop QA, supporting contexts often form a relatively static set of evidence centered around a single question, whereas information in long-term memory tasks changes over time and involves personalization, conflict updates, and context dependence. Therefore, although reader-aware RL feedback can provide transferable writing principles, interaction feedback from the target domain remains crucial for achieving stable performance.

A.3 Writing Protocol and Interaction Budget

Table 7 shows that the writing protocol significantly changes the trade-off among Precision, Recall, and Deducible. The results for Tight=True are 0.836/0.806/0.515; the results for Tight=False are 0.845/0.851/0.506. After relaxing the protocol, Recall improves noticeably, indicating that the writer can write more potentially relevant evidence; however, Deducible declines, suggesting that the additional evidence also contains more noise, redundant facts, or weakly related local information. Although this content may increase the coverage of supporting context, it can dilute the reasoning chain that truly supports the answer.

Iterative writing further highlights the role of the interaction budget. For Iterative, 12 turns, tight, Precision/Recall/Deducible are 0.852/0.829/0.516; after increasing to 20 turns, Recall reaches the highest value of **0.881**, indicating that multi-turn reader feedback helps the writer complete cross-document bridging paths. For Iterative, 24 turns, loose, Precision and Deducible reach **0.863** and **0.531**, respectively, but Recall falls back to 0.826. This shows that more rounds of interaction are not simply "the longer, the better": the benefit comes from the writer revising the graph structure based on reader feedback, whereas when the protocol is too loose or the writing space becomes too large, the additional content may alter

the reader's ranking, causing some gold supporting contexts to be pushed out of the top results.

A.4 Reader-side Sensitivity

Figure 4 analyzes the impact of the frozen reader setting on writer training results. First, the top-k budget sweep shows a non-monotonic trend: at $k = 5$, reward and Deducible are 0.623/0.528, the better setting in this group; at $k = 40$, reward remains at 0.622, but Deducible drops to 0.518; at $k = 60$, reward further declines to 0.591. This indicates that expanding the retrieval budget does not necessarily lead to better reader feedback. Although a larger top-k improves potential coverage, it also introduces more weakly related or redundant evidence, diluting the reasoning chain that truly supports the answer.

The ranker variants also reflect a similar coverage–noise tradeoff. topk20 achieves the highest reward and Deducible, at 0.630/0.544, respectively; idf-only and raw both have a reward of 0.626, but their Deducible scores are 0.538 and 0.519, respectively; idf-topk60 declines to 0.595/0.506. This shows that the reader ranker cannot rely solely on entity overlap or on expanding the candidate set, but instead must strike a balance among entity matching, semantic relevance, and contextual compactness. For the writer, an overly weak ranker makes it difficult for effective graph structure to be read out, while an overly broad candidate space amplifies the negative impact of noisy writes.

Initial-entity weight is the most stable factor among the three groups of reader-side settings. When initial-entity weight is enabled, reward reaches 0.639 and Deducible is 0.525; when it is disabled, reward drops significantly to 0.547 and Deducible falls to 0.505. Even when the budget is increased after disabling it, the rewards for off@10 and off@40 recover only to 0.613 and 0.615. This indicates that the initial entity anchor is crucial in multi-hop graph retrieval: it helps the reader enter the correct local subgraph from the question entity and expand along the bridging relations written by the writer to the evidence supporting the answer. Without this anchor, simply increasing the retrieval budget cannot fully compensate for the deviation in graph traversal direction.

A.5 Training Stability and Regularization

Figure 5 shows the effects of training regularization and rollout settings on the writer. First, the repetition penalty affects both reward and Deducible, but the trend is not monotonic. Without any penalty, the result is 0.585/0.510; with a penalty of 0.10, it improves to 0.610/0.520; with a penalty of 0.50, it reaches the best result in this group at 0.619/0.522; after further increasing it to 1.00, it drops to 0.597/0.500. This indicates that a moderate penalty on repeated triples can suppress redundant edges and cyclic expressions, but an overly strong penalty may limit the writer's necessary restatement of key facts. Especially in multi-hop reasoning, the same bridging entity often needs to appear in multiple relational paths, so repetition is not always meaningless noise.

Rollout filtering brings consistent but limited gains. When filtering is disabled, reward/Deducible is 0.585/0.510; after applying `thr_80`, `thr_90`, and `thr_95`, reward increases to 0.621, 0.617, and 0.625, respectively, while Deducible remains stable at 0.516–0.518. This suggests that filtering out low-quality rollouts can reduce the interference of noisy trajectories with policy updates, allowing the writer to learn effective writing strategies more stably. However, the differences between thresholds are small, indicating that the main role of filtering is to remove obviously negative samples rather than determine the final performance ceiling.

Rollout group size and warmup ratio further affect training stability. As group size increases from $n = 1$ to $n = 10$, reward/Deducible rises from 0.610/0.511 to 0.620/0.531, indicating that a larger group size can provide more reliable relative preference estimates and help RL distinguish more accurately between effective and ineffective writing. The optimal warmup ratio appears at 0.20, where Deducible reaches 0.529; too little warmup may lead to unstable early policy updates, while too much warmup may delay the effect of the RL signal. Overall, these regularization and training scale settings can improve stability, but their gains are smaller than those from reward design, reader-side initial-entity weight, and the interaction protocol itself. This shows that the core improvement of the memory writer comes from reader-aware RL feedback: it forces the graph constructor to learn to preserve bridging entities, cross-document relations, and evidence chains that support answer derivation, while also reducing repetitive structures and irrelevant local facts.

B Signal-to-Noise Ratio and Retrieval Budget of Structurally Gated Propagation

This section analyzes the structural capability of the GFM memory reader in SAGE from the perspective of signal propagation and retrieval budget. Unlike graph-isomorphism expressivity analyses centered on k-WL, we focus on the following question: on noisy graph memories dynamically written by the memory writer, how do soft addressing, structurally gated propagation, context–schema dual-channel calibration, and entity-to-document projection jointly improve the ratio of query-relevant evidence signal to distractor noise, thereby reducing the top-k retrieval budget required to achieve a given level of evidence coverage?

B.1 Review of the SAGE-GFM Reader Formalization

Given a sample $x = (q, D, D^+, y)$, where q denotes the query, $D = \{d_i\}_{i=1}^N$ denotes the candidate memory fragments, and $D^+ \subseteq D$ denotes the gold evidence set supporting the answer y, the memory writer constructs a heterogeneous graph

$$G = W_\theta(q, D) = (V_E \cup V_D, E_{EE} \cup E_{ED}), \tag{6}$$

where V_E is the set of entity nodes, V_D is the set of memory-fragment nodes, E_{EE} denotes entity–entity relation edges, and E_{ED} denotes entity–text-fragment anchoring edges. The GFM memory reader outputs an entity distribution, a document distribution, and an optional activated subgraph:

$$f_\phi(q, G, D) = \big(p_\phi(e \mid q, G),\ p_\phi(d \mid q, G, D),\ G_q\big). \tag{7}$$

The reader first uses query planning and soft addressing to generate query-conditioned initial activation for entities. Let $s_e(q)$ denote the entry score of entity e, which integrates multiple cues such as explicit entities, aliases, pseudo-query similarity, answer type, hard constraints, and entity linking. The initial activation distribution is

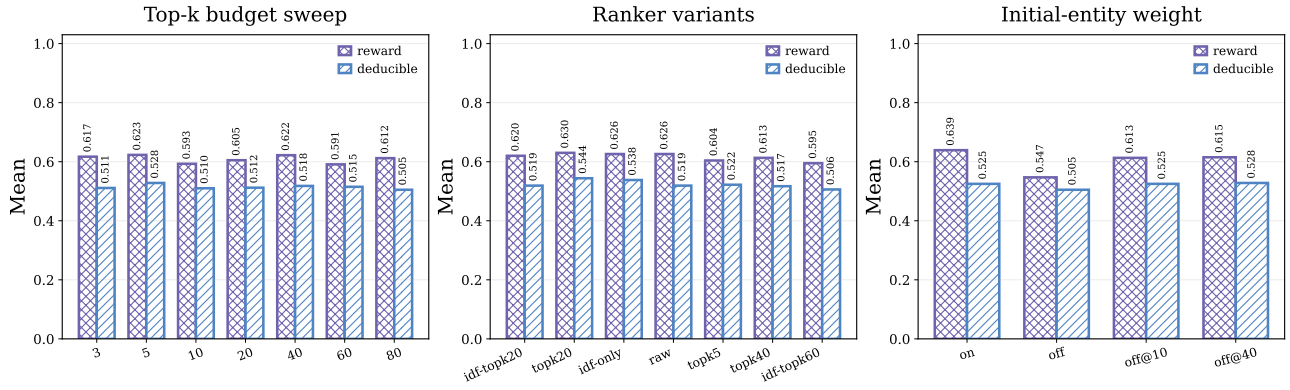

Figure 4: Freeze the sensitivity analysis on the reader side. The impact of the initial-entity weight on reward and Deducible is the most stable; the top-k and ranker variants exhibit a non-monotonic budget–noise trade-off.

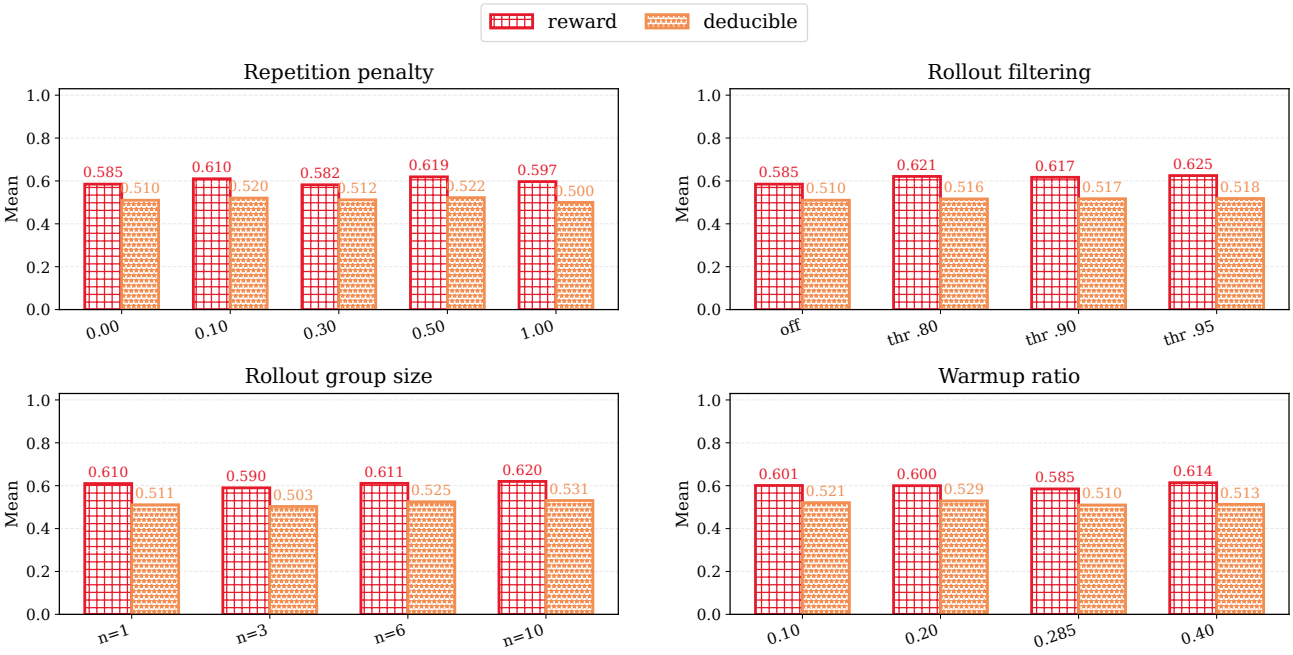

Figure 5: Training regularization and scaling analysis. A larger rollout group size and a moderate warmup ratio show a slight advantage in this batch of results, but the gains are smaller than the effects of reward design, the reader-side initial-entity weight, and the writing protocol.

then given by

$$p_0(e \mid q) = \frac{\exp(s_e(q)/T_0)}{\sum_{v \in V_E} \exp(s_v(q)/T_0)}. \tag{8}$$

The initial entity representation is written as

$$h_e^{(0)} = \big(p_0(e \mid q)\big)^{\eta} W_q \,\mathrm{Emb}(q) + W_x x_e, \qquad 0 \le \eta \le 1. \tag{9}$$

The reader then constructs structural gates using node-level structural features, edge-pair structural features, and graph-level structural summaries. Specifically, let

$$\varphi(v) = \big(\log(1 + d_v),\ c_v,\ \kappa_v,\ \bar{d}_{\mathcal{N}(v)}\big), \tag{10}$$

$$\psi(u, v) = \big(|d_u - d_v|,\ |\mathcal{N}(u) \cap \mathcal{N}(v)|,\ \mathrm{Jaccard}(\mathcal{N}(u), \mathcal{N}(v))\big), \tag{11}$$

$$r_G = \big(\mathrm{mean}_{v \in V_E}\, \varphi(v);\ \mathrm{std}_{v \in V_E}\, \varphi(v);\ \mathrm{dens}(G)\big). \tag{12}$$

The edge structural context at layer l is

$$z_{uv}^{(l)} = \big(E_n^{(l)}(\varphi(u));\ E_n^{(l)}(\varphi(v));\ E_p^{(l)}(\psi(u, v));\ E_g^{(l)}(r_G)\big), \tag{13}$$

which generates the vector-valued gate

$$g_{uv}^{(l)} = 1 + \delta \tanh\left(\mathrm{MLP}_g^{(l)}(z_{uv}^{(l)})\right). \tag{14}$$

Let $\eta_{uv} \geq 0$ denote the normalized adjacency weight with self-loops. The message and node update are

$$m_{u\to v}^{(l)} = \eta_{uv} g_{uv}^{(l)} \odot W_m^{(l)} h_u^{(l-1)}, \tag{15}$$

$$h_v^{(l)} = \mathrm{LayerNorm}\left(h_v^{(l-1)} + \mathrm{PReLU}\left(b^{(l)} + \sum_{u\in\mathcal{N}(v)} m_{u\to v}^{(l)}\right)\right). \tag{16}$$

In addition, the reader combines a contextual calibration channel on the current graph with a cross-graph schema prior channel:

$$H(q, G) = H_{\mathrm{ctx}} + \beta_{\mathrm{sch}} H_{\mathrm{sch}}. \tag{17}$$

B.2 Recoverable Evidence Region and Effective Signal-to-Noise Ratio

Definition B.1 (Recoverable Evidence Region)**.** Fix a query q and the current memory graph G. Let $R_q \subseteq V_E$ denote the recoverable evidence region under query q, namely the set of entities jointly determined by the current graph structure, anchoring edges, and reader-reachable paths. This set contains nodes that support the answer, connect supporting documents, or serve as bridge entities. If the entity anchor set of document d is denoted by $A(d) \subseteq V_E$, then the anchor coverage of the current graph over the gold evidence is defined as

$$\rho_A = \frac{\left|\{d \in D^+ : A(d) \cap R_q \neq \varnothing\}\right|}{|D^+|}. \tag{18}$$

Definition B.2 (Query-Relevant Scalar Activation)**.** Let r_q be the direction induced by the query representation or the final scoring head. The nonnegative query-relevant activation of node v at layer l is defined as

$$a_v^{(l)} = \left[\langle r_q, h_v^{(l)}\rangle\right]_+, \tag{19}$$

where $[t]_+ = \max\{t, 0\}$. The evidence signal mass, noise mass, and effective signal-to-noise ratio at layer l are respectively defined as

$$S_l = \sum_{v\in R_q} a_v^{(l)}, \tag{20}$$

$$N_l = \sum_{v\in V_E\setminus R_q} a_v^{(l)}, \tag{21}$$

$$\mathrm{SNR}_l = \frac{S_l}{N_l}, \tag{22}$$

with the convention that $\mathrm{SNR}_l = +\infty$ when $N_l = 0$.

Remark B.3 (Scope of the Scalar-Channel Analysis). Equation (19) does not assume that the full vector update in Eq. (16) is a nonnegative linear recurrence in every coordinate. LayerNorm, PReLU, residual connections, and linear transformations may all change representation directions. We only analyze the query-relevant channel on which the final retrieval score depends, and absorb the additional effects caused by nonlinearities and directional shifts into a perturbation term. This avoids an overly strong coordinate-wise monotonicity assumption.

B.3 Aggregate Propagation Assumptions and Structural Gating Coefficients

Idealized analyses often assume that every evidence edge has a uniform lower gate bound g_+ and every noisy edge has a uniform upper gate bound g_-. However, in graph memories dynamically constructed by an LLM writer, edges may be missing, erroneous, or repeated; some evidence edges may be underestimated, while some distractor edges may receive high gates. We therefore adopt an aggregate propagation assumption.

Assumption B.4 (Query-Relevant Effective Propagation Operator)**.** Fix a query q, the current graph memory G, and the reader representation at layer l. For each layer $l \in \{1, \ldots, L\}$, there exists a nonnegative matrix $T_l \in \mathbb{R}_{\geq 0}^{|V_E|\times|V_E|}$ and a nonnegative perturbation vector $\epsilon_l \in \mathbb{R}_{\geq 0}^{|V_E|}$ such that the query-relevant activation vector at layer l,

$$a^{(l)} = \left(a_v^{(l)}\right)_{v\in V_E},$$

is controlled by the previous-layer activation $a^{(l-1)}$ in the following coordinate-wise sense:

$$a^{(l)} \preceq T_l a^{(l-1)} + \epsilon_l. \tag{23}$$

Here, $\preceq$ denotes coordinate-wise inequality. The operator T_l denotes the effective propagation operator induced by the l-th layer on the query-relevant scalar channel. It absorbs the combined effects of normalized adjacency weights η_{uv}, structural gates $g_{uv}^{(l)}$, message projection $W_m^{(l)}$, context–schema representation composition, and final scoring-channel projection into a single nonnegative propagation kernel. In other words, $T_l(u, v)$ is the effective nonnegative contribution strength of the query-relevant activation of node v at the previous layer to node u at layer l.

The perturbation term ϵ_l absorbs residual effects that are difficult to exactly characterize by nonnegative linear propagation, including LayerNorm, PReLU, residual connections, vector-direction rotation, scoring-channel mismatch, and finite-parameter approximation error.

Furthermore, we only require the propagation process to preserve effective signal in the evidence region in an aggregate sense. We do not require the structural gate to perfectly distinguish every evidence edge from every noisy edge. The operator T_l may allow some evidence edges to be underestimated and some distractor edges to be overestimated; the aggregate propagation coefficients defined below only characterize the overall effect of these local errors on the evidence and noise regions.

Let

$$\bar{R}_q = V_E \setminus R_q.$$

Partition T_l according to the node sets R_q and $\bar{R}_q$:

$$T_l = \begin{pmatrix} T_{RR}^{(l)} & T_{R\bar{R}}^{(l)} \\ T_{\bar{R}R}^{(l)} & T_{\bar{R}\bar{R}}^{(l)} \end{pmatrix}. \tag{24}$$

Here, $T_{RR}^{(l)}$ denotes effective propagation within the evidence region, $T_{\bar{R}\bar{R}}^{(l)}$ denotes effective propagation within the noise region, $T_{\bar{R}R}^{(l)}$ denotes leakage propagation from the evidence region to the noise region, and $T_{R\bar{R}}^{(l)}$ denotes propagation from the noise region to the evidence region.

Definition B.5 (Aggregate Propagation Coefficients)**.** Given the effective propagation operator T_l at layer l and its block decomposition in Eq. (24), define the three aggregate propagation coefficients A_l, B_l, and C_l as follows.

A_l is the evidence-retention coefficient. It characterizes the minimum fraction of total mass that remains inside the evidence region R_q after any nonnegative evidence signal x propagates one layer within R_q. Formally, A_l is any nonnegative constant satisfying

$$A_l \le \inf_{x\in\mathbb{R}_{\ge 0}^{|R_q|},\ \mathbf{1}^\top x>0} \frac{\mathbf{1}^\top T_{RR}^{(l)} x}{\mathbf{1}^\top x}. \tag{25}$$

Equivalently, for any nonnegative evidence signal $x \in \mathbb{R}_{\ge 0}^{|R_q|}$ with $\mathbf{1}^\top x > 0$,

$$\mathbf{1}^\top T_{RR}^{(l)} x \ge A_l\, \mathbf{1}^\top x.$$

B_l is the noise self-propagation coefficient. It characterizes the maximum extent to which any nonnegative noise signal y can be retained or expanded after one layer of propagation inside the noise region $\bar{R}_q$. Formally, B_l is any nonnegative constant satisfying

$$B_l \ge \sup_{y\in\mathbb{R}_{\ge 0}^{|\bar{R}_q|},\ \mathbf{1}^\top y>0} \frac{\mathbf{1}^\top T_{\bar{R}\bar{R}}^{(l)} y}{\mathbf{1}^\top y}. \tag{26}$$

Equivalently, for any nonnegative noise signal $y \in \mathbb{R}_{\ge 0}^{|\bar{R}_q|}$ with $\mathbf{1}^\top y > 0$,

$$\mathbf{1}^\top T_{\bar{R}\bar{R}}^{(l)} y \le B_l\, \mathbf{1}^\top y.$$

C_l is the evidence-to-noise leakage coefficient. It characterizes the maximum fraction of an arbitrary nonnegative signal in the evidence region that can leak into the non-evidence region $\bar{R}_q$ after one layer of propagation. Formally, C_l is any nonnegative constant satisfying

$$C_l \ge \sup_{x\in\mathbb{R}_{\ge 0}^{|R_q|},\ \mathbf{1}^\top x>0} \frac{\mathbf{1}^\top T_{\bar{R}R}^{(l)} x}{\mathbf{1}^\top x}. \tag{27}$$

Equivalently, for any nonnegative evidence signal $x \in \mathbb{R}_{\ge 0}^{|R_q|}$ with $\mathbf{1}^\top x > 0$,

$$\mathbf{1}^\top T_{\bar{R}R}^{(l)} x \le C_l\, \mathbf{1}^\top x.$$

Finally, let

$$\xi_l = \mathbf{1}^\top \epsilon_{l,\bar{R}} \tag{28}$$

denote the total perturbation mass injected into the noise region $\bar{R}_q$ at layer l by nonlinearities, normalization, representation-direction shifts, and approximation errors. Here, $\epsilon_{l,\bar{R}}$ denotes the restriction of the perturbation vector ϵ_l to $\bar{R}_q$.

Lemma B.6 (Aggregate Propagation Recurrence)**.** *Under Assumption B.4 and Definition B.5, if $S_{l-1} > 0$ and $N_{l-1} \ge 0$, then layer l satisfies*

$$S_l \ge A_l S_{l-1}, \tag{29}$$

$$N_l \le B_l N_{l-1} + C_l S_{l-1} + \xi_l. \tag{30}$$

Proof. Let $a_R^{(l-1)}$ and $a_{\bar{R}}^{(l-1)}$ be the restrictions of $a^{(l-1)}$ to R_q and $\bar{R}_q$, respectively. By the definition of the evidence-retention coefficient in Eq. (25), the total mass retained within the evidence region through $R_q \to R_q$ propagation is at least

$$\mathbf{1}^\top T_{RR}^{(l)} a_R^{(l-1)} \ge A_l \mathbf{1}^\top a_R^{(l-1)} = A_l S_{l-1}, \tag{31}$$

and thus $S_l \ge A_l S_{l-1}$.

On the other hand, the layer-l mass in the noise region can be upper-bounded by three terms: noise self-propagation, evidence leakage, and perturbation:

$$N_l \le \mathbf{1}^\top T_{\bar{R}\bar{R}}^{(l)} a_{\bar{R}}^{(l-1)} + \mathbf{1}^\top T_{\bar{R}R}^{(l)} a_R^{(l-1)} + \mathbf{1}^\top \epsilon_{l,\bar{R}}. \tag{32}$$

Using Eqs. (26), (27), and (28), we obtain

$$N_l \le B_l N_{l-1} + C_l S_{l-1} + \xi_l. \tag{33}$$

This proves the lemma. □

B.4 Realistic Aggregate Signal-to-Noise Ratio Bound

Theorem B.7 (Realistic Aggregate SNR Bound)**.** *Assume that for all $l = 1, \dots, L$, there exist $A_l > 0$, $B_l \ge 0$, $C_l \ge 0$, and $\xi_l \ge 0$ such that the recurrences in Eqs.* (29)–(30) *hold. Let*

$$Q_l = \mathrm{SNR}_l^{-1} = \frac{N_l}{S_l}. \tag{34}$$

Then

$$Q_L \le \left(\prod_{l=1}^{L} \frac{B_l}{A_l}\right) Q_0 + \sum_{i=1}^{L} \left(\frac{C_i}{A_i} + \frac{\xi_i}{A_i S_{i-1}}\right) \prod_{t=i+1}^{L} \frac{B_t}{A_t}. \tag{35}$$

Equivalently, if the right-hand side is finite, then

$$\mathrm{SNR}_L \ge \left[\left(\prod_{l=1}^{L} \frac{B_l}{A_l}\right) \mathrm{SNR}_0^{-1} + \sum_{i=1}^{L} \left(\frac{C_i}{A_i} + \frac{\xi_i}{A_i S_{i-1}}\right) \prod_{t=i+1}^{L} \frac{B_t}{A_t}\right]^{-1}. \tag{36}$$

The empty product is defined as 1.

Proof. By Lemma B.6, for any l,

$$S_l \ge A_l S_{l-1}, \qquad N_l \le B_l N_{l-1} + C_l S_{l-1} + \xi_l. \tag{37}$$

Therefore,

$$Q_l = \frac{N_l}{S_l} \le \frac{B_l N_{l-1} + C_l S_{l-1} + \xi_l}{A_l S_{l-1}} \tag{38}$$

$$= \frac{B_l}{A_l} Q_{l-1} + \frac{C_l}{A_l} + \frac{\xi_l}{A_l S_{l-1}}. \tag{39}$$

Let

$$r_l = \frac{B_l}{A_l}, \qquad d_l = \frac{C_l}{A_l} + \frac{\xi_l}{A_l S_{l-1}}. \tag{40}$$

Then

$$Q_l \le r_l Q_{l-1} + d_l. \tag{41}$$

Expanding this first-order nonhomogeneous recurrence yields

$$Q_L \le \left(\prod_{l=1}^{L} r_l\right) Q_0 + \sum_{i=1}^{L} d_i \prod_{t=i+1}^{L} r_t. \tag{42}$$

Substituting back r_l and d_l proves Eq. (35). Since $\mathrm{SNR}_L = Q_L^{-1}$, Eq. (36) follows. □

Corollary B.8 (Layer-Homogeneous Case)**.** *If $A_l = A > 0$, $B_l = B \ge 0$, $C_l = C \ge 0$, and $\xi_l = 0$, then*

$$\mathrm{SNR}_L \ge \left[\left(\frac{B}{A}\right)^L \mathrm{SNR}_0^{-1} + \frac{C}{A} \sum_{i=0}^{L-1} \left(\frac{B}{A}\right)^i\right]^{-1}. \tag{43}$$

If further $C = 0$, then

$$\mathrm{SNR}_L \ge \left(\frac{A}{B}\right)^L \mathrm{SNR}_0\,. \tag{44}$$

Proof. Substituting $A_l = A$, $B_l = B$, $C_l = C$, and $\xi_l = 0$ into Theorem B.7 and simplifying the resulting geometric series gives the result. □

Corollary B.9 (Ideal Edge-Wise Gating as a Special Case). *Suppose there exist constants $g_+ > g_- > 0$, $\alpha_+ > 0$, $\alpha_- > 0$, $g_0 \geq 0$, and $\lambda_{\text{leak}} \geq 0$ such that the effective retention inside the evidence region is $A = g_+\alpha_+$, the self-propagation inside the noise region is $B = g_-\alpha_-$, the evidence-to-noise leakage is $C = g_0\lambda_{\text{leak}}$, and $\xi_l = 0$. Then Theorem B.7 reduces to*

$$\mathrm{SNR}_L \geq \left[\left(\frac{g_-\alpha_-}{g_+\alpha_+}\right)^L \mathrm{SNR}_0^{-1} + \frac{g_0\lambda_{\text{leak}}}{g_+\alpha_+}\sum_{i=0}^{L-1}\left(\frac{g_-\alpha_-}{g_+\alpha_+}\right)^i\right]^{-1}. \tag{45}$$

If $\lambda_{\text{leak}} = 0$, then

$$\mathrm{SNR}_L \geq \left(\frac{g_+\alpha_+}{g_-\alpha_-}\right)^L \mathrm{SNR}_0\,. \tag{46}$$

B.5 Document Retrieval Budget

The final retrieval targets of the SAGE-GFM reader are memory fragments or documents. Therefore, we need to convert the entity-level SNR bound into a document-level top-k budget bound. Let the final document score be $S_D(d)$, and let the top-k retrieval result be

$$P_k(q, G) = \text{Top-}k_{d\in D} S_D(d). \tag{47}$$

Definition B.10 (ρ-Coverage Retrieval Budget). Given $0 < \rho \leq \rho_A$, let

$$m_\rho = \lceil \rho |D^+| \rceil. \tag{48}$$

The minimum top-k budget required to achieve ρ-level gold evidence coverage is defined as

$$\mathcal{B}_\rho(q, G) = \min\left\{k : \ |P_k(q, G) \cap D^+| \geq m_\rho\right\}. \tag{49}$$

Let τ_ρ^+ denote the m_ρ-th largest score among gold evidence documents, namely the gold score threshold required to achieve ρ-coverage.

Lemma B.11 (Quantile Retrieval Budget Bound). *Let the total score mass of distractor documents be*

$$M_L^- = \sum_{d\in D\setminus D^+} S_D(d). \tag{50}$$

If $\tau_\rho^+ > 0$, then

$$\mathcal{B}_\rho(q, G) \leq m_\rho + \frac{M_L^-}{\tau_\rho^+}. \tag{51}$$

Proof. Define the set of distractor documents whose scores are not lower than τ_ρ^+ as

$$\mathcal{N}_\rho = \{d \in D \setminus D^+ : S_D(d) \geq \tau_\rho^+\}. \tag{52}$$

For any $d \in \mathcal{N}_\rho$, we have $S_D(d) \geq \tau_\rho^+$, and hence

$$|\mathcal{N}_\rho|\tau_\rho^+ \leq \sum_{d\in\mathcal{N}_\rho} S_D(d) \leq M_L^-. \tag{53}$$

Thus $|\mathcal{N}_\rho| \leq M_L^-/\tau_\rho^+$. To ensure that the top-k results contain at least m_ρ gold evidence documents, it suffices to include these m_ρ gold documents and all distractors whose scores are not lower than the threshold τ_ρ^+. Therefore,

$$\mathcal{B}_\rho(q, G) \leq m_\rho + |\mathcal{N}_\rho| \leq m_\rho + \frac{M_L^-}{\tau_\rho^+}. \tag{54}$$

This proves the lemma. □

To use entity-level SNR for document-level retrieval, we need to control the noise expansion introduced by entity-to-document projection.

Assumption B.12 (Projection Noise and Gold Score Concentration). There exist constants $K_A \geq 0$, $\zeta_A \geq 0$, and $c_\rho \in (0, 1]$ such that the final document scores satisfy

$$M_L^- \leq K_A N_L + \zeta_A, \tag{55}$$

$$\tau_\rho^+ \geq \frac{c_\rho}{m_\rho} S_L. \tag{56}$$

Here, K_A is the noise expansion factor of entity-to-document projection, ζ_A denotes the projection residual caused by incorrect anchors, missing anchors, or additional text-similarity terms, and c_ρ measures whether the evidence signal is effectively distributed over at least m_ρ gold documents.

Theorem B.13 (Realistic Signal–Noise–Budget Bound). *Under the conditions of Theorem B.7, further assume that Assumption B.12 holds. Then*

$$\mathcal{B}_\rho(q, G) \leq m_\rho + \frac{m_\rho K_A}{c_\rho}\mathrm{SNR}_L^{-1} + \frac{m_\rho\zeta_A}{c_\rho S_L}. \tag{57}$$

Substituting Theorem B.7 further yields the explicit upper bound

$$\begin{aligned}\mathcal{B}_\rho(q, G) \leq m_\rho + \frac{m_\rho K_A}{c_\rho}\Bigg[&\left(\prod_{l=1}^L \frac{B_l}{A_l}\right)\mathrm{SNR}_0^{-1} \\ &+ \sum_{i=1}^L\left(\frac{C_i}{A_i} + \frac{\xi_i}{A_i S_{i-1}}\right)\prod_{t=i+1}^L \frac{B_t}{A_t}\Bigg] + \frac{m_\rho\zeta_A}{c_\rho S_L}.\end{aligned} \tag{58}$$

Proof. By Lemma B.11,

$$\mathcal{B}_\rho(q, G) \leq m_\rho + \frac{M_L^-}{\tau_\rho^+}. \tag{59}$$

By Assumption B.12,

$$\frac{M_L^-}{\tau_\rho^+} \leq \frac{K_A N_L + \zeta_A}{(c_\rho/m_\rho)S_L} = \frac{m_\rho K_A}{c_\rho}\frac{N_L}{S_L} + \frac{m_\rho\zeta_A}{c_\rho S_L}. \tag{60}$$

Since $N_L/S_L = \mathrm{SNR}_L^{-1}$, Eq. (57) follows. Substituting the upper bound on SNR_L^{-1} from Theorem B.7 into Eq. (57) gives Eq. (58). □

Corollary B.14 (Full Evidence Recovery Budget). *If $\rho = 1$ and $\rho_A = 1$, then $m_\rho = |D^+|$, and $\mathcal{B}_\rho(q, G)$ reduces to the full evidence recovery budget. In this case, Theorem B.13 provides an upper bound on the top-k budget required to recover all gold evidence.*

B.6 Interpretation of the Theoretical Bound for the SAGE Design

Theorem B.7 and Theorem B.13 unify four reader design factors under the same retrieval-budget upper bound. To avoid relying only on intuitive discussion, we provide several direct monotonicity propositions.

Proposition 2 (Monotonicity of the Budget Bound). Fix m_ρ, K_A, c_ρ, ζ_A, S_L, and SNR_0, and define

$$\Gamma_L = \left(\prod_{l=1}^L \frac{B_l}{A_l}\right)\mathrm{SNR}_0^{-1} + \sum_{i=1}^L\left(\frac{C_i}{A_i} + \frac{\xi_i}{A_i S_{i-1}}\right)\prod_{t=i+1}^L \frac{B_t}{A_t}. \tag{61}$$

Then the budget upper bound

$$U_\rho = m_\rho + \frac{m_\rho K_A}{c_\rho}\Gamma_L + \frac{m_\rho \zeta_A}{c_\rho S_L} \tag{62}$$

is monotonically nondecreasing in Γ_L, K_A, and ζ_A, and monotonically nonincreasing in c_ρ and S_L. If all other terms in the products are fixed, decreasing any of B_l/A_l, C_l/A_l, or $\xi_l/(A_l S_{l-1})$ cannot increase U_ρ.

Proof. The partial derivatives of U_ρ with respect to Γ_L, K_A, and ζ_A are nonnegative, while the partial derivatives with respect to c_ρ and S_L are nonpositive. Moreover, Γ_L is a nonnegative linear or multiplicative combination of B_l/A_l, C_l/A_l, and $\xi_l/(A_l S_{l-1})$. When the other terms are fixed, decreasing any such nonnegative term cannot increase Γ_L. The proposition follows. □

Proposition 3 (Effect of Soft Addressing). If soft addressing increases the initial evidence signal S_0 and decreases the initial noise mass N_0, thereby increasing SNR_0, then, with all other coefficients fixed, the budget upper bound in Theorem B.13 does not increase. In particular, explicit entities, aliases, pseudo-queries, type constraints, hard constraints, and entity-linking signals in query planning improve the final budget bound whenever they increase S_0/N_0 in an aggregate sense.

Proposition 4 (Aggregate Advantage of Structural Gating). Compared with a reader without structural gating, suppose the structurally gated reader satisfies

$$\frac{B_l^{\mathrm{gate}}}{A_l^{\mathrm{gate}}} \le \frac{B_l^{\mathrm{plain}}}{A_l^{\mathrm{plain}}}, \qquad \frac{C_l^{\mathrm{gate}}}{A_l^{\mathrm{gate}}} \le \frac{C_l^{\mathrm{plain}}}{A_l^{\mathrm{plain}}}, \qquad \frac{\xi_l^{\mathrm{gate}}}{A_l^{\mathrm{gate}} S_{l-1}^{\mathrm{gate}}} \le \frac{\xi_l^{\mathrm{plain}}}{A_l^{\mathrm{plain}} S_{l-1}^{\mathrm{plain}}}. \tag{63}$$

Then the budget upper bound of the structurally gated reader is no larger than that of the ungated reader.

Proposition 5 (Stability Interpretation of the Context–Schema Dual Channel). Let δ_l denote the failure probability of the aggregate recurrences in Eqs. (29)–(30) at layer l. If the schema prior channel reduces the variance of cross-graph structural-role estimation and the context calibration channel reduces the current-graph adaptation error, so that δ_l decreases to δ_l' with $\delta_l' \le \delta_l$, then the probability lower bound under which Theorem B.7 and Theorem B.13 simultaneously hold improves from $1 - \sum_{l=1}^{L} \delta_l$ to $1 - \sum_{l=1}^{L} \delta_l'$.

The core quantities derived above are

$$\mathrm{SNR}_L^{-1} \le \left(\prod_{l=1}^{L} \frac{B_l}{A_l}\right) \mathrm{SNR}_0^{-1} + \sum_{i=1}^{L} \left(\frac{C_i}{A_i} + \frac{\xi_i}{A_i S_{i-1}}\right) \prod_{t=i+1}^{L} \frac{B_t}{A_t}, \tag{64}$$

and

$$\mathcal{B}_\rho(q, G) \le m_\rho + \frac{m_\rho K_A}{c_\rho}\, \mathrm{SNR}_L^{-1} + \frac{m_\rho \zeta_A}{c_\rho S_L}. \tag{65}$$

Equation (64) shows that soft addressing reduces the amount of noise that subsequent propagation must overcome by improving the initial SNR_0; structural gating improves aggregate evidence retention and noise suppression by increasing A_l and decreasing B_l and C_l; the context–schema dual channel makes these aggregate inequalities more stable on dynamic graph memories by reducing cross-graph structural-role estimation error; and entity-to-document projection converts entity-level SNR into document-level budget efficiency by decreasing K_A and ζ_A and increasing c_ρ. Equation (65) further shows that the advantage of SAGE-GFM does not rely on perfect edge-wise classification or zero-leakage assumptions. As long as evidence-retention dominance is achieved in an aggregate or high-probability sense, i.e., B_l/A_l and C_l/A_l are sufficiently small, the reader can improve query-relevant SNR and reduce the top-k retrieval budget required to achieve a given level of evidence coverage.

C Target Graph Calibration and Cross-graph Structural Priors

C.1 Structural Role Decomposition Assumption

Definition C.1 (Structural role mapping)**.** Given a graph G, a mapping

$$\rho_G : V(G) \to \mathcal{R} \tag{66}$$

is called a structural role mapping, where $\mathcal{R}$ is the structural role space. $\rho_G(v)$ can be jointly determined by $\varphi_G(v)$, the structural statistics of edges incident to v, local community-boundary statistics, and other graph-structure summaries. Typical structural roles include hub, bridge, community core, boundary node, and noisy shortcut.

Definition C.2 (Target graph reading risk)**.** Fix a target graph G. Let $\mathcal{D}_G$ be the query–node sampling distribution on the target graph, and let $f_G^\star(q, v)$ be the ideal evidence relevance function. For any measurable function f, define the squared risk as

$$\mathcal{R}_G(f) = \mathbb{E}_{(q,v)\sim\mathcal{D}_G}\left[\left(f(q, v, G) - f_G^\star(q, v)\right)^2\right]. \tag{67}$$

Assumption C.3 (Context–schema decomposability)**.** For every target graph G, the ideal reading function can be decomposed as

$$f_G^\star(q, v) = f_{\mathrm{sch}}^\star(q, \rho_G(v)) + f_{\mathrm{ctx},G}^\star(q, v), \tag{68}$$

where $f_{\mathrm{sch}}^\star$ denotes the cross-graph shared structural reading rule, and $f_{\mathrm{ctx},G}^\star$ denotes the target-graph residual induced by the current writer, current domain, current entity naming, local noise, and writing style.

Equation (68) corresponds exactly to the structural design of SAGE: H_{sch} is used to approximate $f_{\mathrm{sch}}^\star$, and H_{ctx} is used to approximate $f_{\mathrm{ctx},G}^\star$. The next subsection gives the risk meaning of this decomposition.

C.2 Approximation Risk of Context–schema Decomposition

Theorem C.4 (Context–schema decomposition reduces target–graph approximation risk)**.** *Suppose Assumption 2.3 holds. Let $\mathcal{H}_{\mathrm{sch}}$ be the schema function class, let $\mathcal{H}_{\mathrm{ctx},G}$ be the target-graph context function class, and define the sum class*

$$\mathcal{H}_{\mathrm{sch}} + \mathcal{H}_{\mathrm{ctx},G} = \{f_s + f_c : f_s \in \mathcal{H}_{\mathrm{sch}}, f_c \in \mathcal{H}_{\mathrm{ctx},G}\}. \tag{69}$$

If there exist $\epsilon_{\mathrm{sch}}, \epsilon_{\mathrm{ctx}} \ge 0$ such that

$$\inf_{f_s \in \mathcal{H}_{\mathrm{sch}}} \mathbb{E}\left[\left(f_s(q, \rho_G(v)) - f_{\mathrm{sch}}^\star(q, \rho_G(v))\right)^2\right] \le \epsilon_{\mathrm{sch}}, \tag{70}$$

$$\inf_{f_c \in \mathcal{H}_{\mathrm{ctx},G}} \mathbb{E}\left[\left(f_c(q, v, G) - f_{\mathrm{ctx},G}^\star(q, v)\right)^2\right] \le \epsilon_{\mathrm{ctx}}, \tag{71}$$

where both expectations are over $(q, v) \sim \mathcal{D}_G$, then

$$\inf_{f \in \mathcal{H}_{\text{sch}} + \mathcal{H}_{\text{ctx},G}} \mathcal{R}_G(f) \le 2\epsilon_{\text{sch}} + 2\epsilon_{\text{ctx}}. \tag{72}$$

Proof. Take any $\alpha > 0$. By the two approximation error conditions, there exist $\hat{f}_s \in \mathcal{H}_{\text{sch}}$ and $\hat{f}_c \in \mathcal{H}_{\text{ctx},G}$ such that

$$\mathbb{E}[(\hat{f}_s - f^\star_{\text{sch}})^2] \le \epsilon_{\text{sch}} + \alpha, \qquad \mathbb{E}[(\hat{f}_c - f^\star_{\text{ctx},G})^2] \le \epsilon_{\text{ctx}} + \alpha. \tag{73}$$

Let $\hat{f} = \hat{f}_s + \hat{f}_c$. By the decomposition in Eq. (68), we have

$$\hat{f} - f^\star_G = (\hat{f}_s - f^\star_{\text{sch}}) + (\hat{f}_c - f^\star_{\text{ctx},G}). \tag{74}$$

Using $(a+b)^2 \le 2a^2 + 2b^2$, we obtain

$$\begin{aligned} \mathcal{R}_G(\hat{f}) &= \mathbb{E}[(\hat{f} - f^\star_G)^2] \\ &\le 2\mathbb{E}[(\hat{f}_s - f^\star_{\text{sch}})^2] + 2\mathbb{E}[(\hat{f}_c - f^\star_{\text{ctx},G})^2] \\ &\le 2\epsilon_{\text{sch}} + 2\epsilon_{\text{ctx}} + 4\alpha. \end{aligned} \tag{75}$$

Since $\alpha > 0$ is arbitrary, taking the infimum yields the conclusion. □

Proposition 6 (Residual bias of schema-only models). Further assume that $L_2(\mathcal{D}_G)$ is a Hilbert space, $\mathcal{H}_{\text{sch}}$ is a closed linear subspace of it, and $f^\star_{\text{sch}} \in \mathcal{H}_{\text{sch}}$. If only a schema-only model $f_s \in \mathcal{H}_{\text{sch}}$ is used, then

$$\inf_{f_s \in \mathcal{H}_{\text{sch}}} \mathcal{R}_G(f_s) = \text{dist}^2_{L_2(\mathcal{D}_G)}(f^\star_{\text{ctx},G}, \mathcal{H}_{\text{sch}}). \tag{76}$$

Therefore, as long as the target-graph residual $f^\star_{\text{ctx},G}$ does not belong to $\mathcal{H}_{\text{sch}}$, a schema-only reader has an irreducible target-graph bias.

Proof. By $f^\star_{\text{sch}} \in \mathcal{H}_{\text{sch}}$ and the linearity of $\mathcal{H}_{\text{sch}}$, any $f_s \in \mathcal{H}_{\text{sch}}$ can be written as $f_s = f^\star_{\text{sch}} + g$, where $g \in \mathcal{H}_{\text{sch}}$. Thus,

$$\begin{aligned} \mathcal{R}_G(f_s) &= \left\| f_s - f^\star_{\text{sch}} - f^\star_{\text{ctx},G} \right\|^2_{L_2(\mathcal{D}_G)} \\ &= \left\| g - f^\star_{\text{ctx},G} \right\|^2_{L_2(\mathcal{D}_G)}. \end{aligned} \tag{77}$$

Taking the infimum over $f_s \in \mathcal{H}_{\text{sch}}$ is equivalent to taking the infimum over $g \in \mathcal{H}_{\text{sch}}$, and Eq. (76) follows from the definition of distance. □

Remark C.5. Proposition 2.5 shows that the cross-graph schema prior can only characterize cross-graph shared structural rules and cannot replace target graph calibration. In contrast, the role of the target graph calibration channel H_{ctx} is to absorb $f^\star_{\text{ctx},G}$, namely the local noise, entity granularity, relation style, and domain residual of the graph generated by the current writer.

C.3 Sample Complexity Advantage of Schema Prior

Lemma C.6 (Uniform convergence for bounded loss classes). *Let $\mathcal{F}$ be a function class, and let the loss $\ell_f(z) \in [0,1]$. Given n independent samples $S = \{z_i\}_{i=1}^n$, define the true risk $R(f) = \mathbb{E}[\ell_f(z)]$ and the empirical risk $\widehat{R}_S(f) = n^{-1}\sum_{i=1}^n \ell_f(z_i)$. Then, with probability at least $1-\delta$, for all $f \in \mathcal{F}$ simultaneously,*

$$R(f) \le \widehat{R}_S(f) + 2\,\text{Rad}_n(\ell \circ \mathcal{F}) + 3\sqrt{\frac{\log(2/\delta)}{2n}}, \tag{78}$$

where $\text{Rad}_n(\ell \circ \mathcal{F})$ is an upper bound on the empirical Rademacher complexity of the loss-composed class.

Proof. This is a direct result of the standard symmetrization and McDiarmid/Hoeffding concentration inequalities for bounded loss function classes. Specifically, let

$$Z(S) = \sup_{f \in \mathcal{F}} \left| R(f) - \widehat{R}_S(f) \right|. \tag{79}$$

By symmetrization, $\mathbb{E}Z(S) \le 2\,\text{Rad}_n(\ell \circ \mathcal{F})$. Moreover, since changing one sample can change $Z(S)$ by at most $1/n$, McDiarmid's inequality gives

$$Z(S) \le \mathbb{E}Z(S) + 3\sqrt{\frac{\log(2/\delta)}{2n}} \tag{80}$$

with probability at least $1-\delta$. Combining the two inequalities gives the conclusion. □

Theorem C.7 (Schema prior reduces the sample complexity of target-graph adaptation). *Fix a target graph G, and suppose the number of supervised samples available for reader calibration on the target graph is n_G. Let $\mathcal{H}_{\text{full}}$ be the full reader class that needs to be learned on the target graph when no schema prior is used; let $\mathcal{H}_{\text{res}}$ be the residual class that only needs to be learned given a schema prior f_s, with the combined model $f = f_s + f_r$, $f_r \in \mathcal{H}_{\text{res}}$. Let $\hat{f}_{\text{full}}$ and $\hat{f}_{\text{res}}$ be the empirical risk minimizers over the two classes, respectively. Then, with probability at least $1-\delta$,*

$$\mathcal{R}_G(\hat{f}_{\text{full}}) \le \inf_{f \in \mathcal{H}_{\text{full}}} \mathcal{R}_G(f) + 4\,\text{Rad}_{n_G}(\ell \circ \mathcal{H}_{\text{full}}) + 6\sqrt{\frac{\log(4/\delta)}{2n_G}}, \tag{81}$$

$$\mathcal{R}_G(f_s + \hat{f}_{\text{res}}) \le \inf_{f_r \in \mathcal{H}_{\text{res}}} \mathcal{R}_G(f_s + f_r) + 4\,\text{Rad}_{n_G}(\ell \circ (f_s + \mathcal{H}_{\text{res}})) \tag{82}$$

$$+ 6\sqrt{\frac{\log(4/\delta)}{2n_G}}. \tag{83}$$

If the complexities satisfy

$$\text{Rad}_{n_G}(\ell \circ \mathcal{H}_{\text{full}}) = \widetilde{O}\left(\sqrt{\frac{d_{\text{full}}}{n_G}}\right), \qquad \text{Rad}_{n_G}(\ell \circ (f_s + \mathcal{H}_{\text{res}})) = \widetilde{O}\left(\sqrt{\frac{d_{\text{res}}}{n_G}}\right), \tag{84}$$

and $d_{\text{res}} \ll d_{\text{full}}$, then the schema prior reduces target-graph learning from estimating the full reading function to estimating the residual, and lowers the estimation error term for target-graph adaptation.

Proof. For any function class $\mathcal{F}$, let $\hat{f}$ be the empirical risk minimizer and let $f^\circ \in \arg\inf_{f \in \mathcal{F}} R(f)$ be the true risk minimizer. By empirical optimality, $\widehat{R}(\hat{f}) \le \widehat{R}(f^\circ)$, so

$$\begin{aligned} R(\hat{f}) - R(f^\circ) &= R(\hat{f}) - \widehat{R}(\hat{f}) + \widehat{R}(\hat{f}) - \widehat{R}(f^\circ) + \widehat{R}(f^\circ) - R(f^\circ) \\ &\le 2 \sup_{f \in \mathcal{F}} \left| R(f) - \widehat{R}(f) \right|. \end{aligned} \tag{85}$$

Applying Lemma 2.7 to $\mathcal{F} = \mathcal{H}_{\text{full}}$ and $\mathcal{F} = f_s + \mathcal{H}_{\text{res}}$, respectively, and combining the two probability events by a union bound, gives Eq. (81) and Eq. (83). The complexity-order conclusion follows by substituting the corresponding Rademacher upper bounds. □

Remark C.8. Theorem 2.8 gives the key theoretical motivation for the schema prior: in self-evolving memory, each new graph generated by the writer usually provides only limited supervised

signal. If the reader relearns the full reading function from the target graph in every round, high-variance estimation arises; the schema prior fixes or strongly regularizes the cross-graph shared structural rules, so that target-graph calibration only needs to learn the residual, thereby reducing sample complexity.

D Writer-induced Graph Distribution Shift and Target Graph Calibration

D.1 Writer-induced Dynamic Graph Distribution

The writer parameter θ induces a graph distribution $P_\theta(G \mid q, D)$ on the sample (q, D). For notational simplicity, denote the joint distribution of (q, D, D^+, y, G) by Π_θ. Given the reader parameter ϕ, define the reader risk as

$$\mathcal{L}_R(\phi;\theta) = \mathbb{E}_{(q,D,D^+,y,G)\sim\Pi_\theta}\left[\ell_R(R_\phi(q,G,D), D^+, y)\right]. \tag{86}$$

Here, ℓ_R can be supporting-entity BCE, multi-positive ranking loss, document recall loss, or a combination thereof.

Proposition 7 (Writer updates cause reader distribution shift). Assume $0 \le \ell_R \le 1$. For any fixed ϕ and any writer parameters θ, θ', we have

$$|\mathcal{L}_R(\phi;\theta') - \mathcal{L}_R(\phi;\theta)| \le \mathrm{TV}(\Pi_{\theta'}, \Pi_\theta), \tag{87}$$

where TV is the total variation distance. If, further, $\ell_R(R_\phi(q,G,D), D^+, y)$ is L_ℓ-Lipschitz with respect to the graph variable under some graph metric $d_{\mathcal{G}}$, then

$$|\mathcal{L}_R(\phi;\theta') - \mathcal{L}_R(\phi;\theta)| \le L_\ell W_1(\Pi_{\theta'}, \Pi_\theta), \tag{88}$$

where W_1 is the first-order Wasserstein distance induced by $d_{\mathcal{G}}$.

Proof. For the bounded loss case, let $h_\phi(q, D, D^+, y, G) = \ell_R(R_\phi(q,G,D), D^+, y) \in [0,1]$. Then

$$|\mathcal{L}_R(\phi;\theta') - \mathcal{L}_R(\phi;\theta)| = \left|\int h_\phi \, d\Pi_{\theta'} - \int h_\phi \, d\Pi_\theta\right| \le \mathrm{TV}(\Pi_{\theta'}, \Pi_\theta), \tag{89}$$

where the last step follows from the dual definition of total variation. If h_ϕ is L_ℓ-Lipschitz, the Wasserstein upper bound follows from the Kantorovich–Rubinstein duality. □

Corollary D.1 (Necessity of target graph calibration). *If updating the writer from θ to θ' causes $\mathrm{TV}(\Pi_{\theta'}, \Pi_\theta)$ to be non-negligible, then the risk of a fixed reader ϕ on the new graph distribution may increase. Therefore, target graph calibration of the reader, namely $\phi \mapsto \phi'$ to reduce $\mathcal{L}_R(\phi';\theta')$, is a necessary mechanism for handling writer-induced graph distribution shift.*

Proof. By Proposition 3.1, writer distribution shift can directly change the new-distribution risk of the fixed reader. If the reader is not updated, there is no optimization mechanism to offset this drift term. Target graph calibration is precisely re-optimization of $\mathcal{L}_R(\cdot;\theta')$, and is therefore a natural step for reducing the risk on the new graph. □

E Reader Stability under Dynamic Graph Evolution

E.1 Realistic Graph Evolution Distance

Real graphs often contain hubs, node additions and deletions, alias merges, anchor rewrites, and structural statistics that are not globally Lipschitz. Therefore, we use the augmented graph drift actually perceived by the reader to measure graph evolution.

Definition E.1 (Padding alignment and presence bit). Given two consecutive-round graphs G and G', align them through persistent memory ids to a common node universe $\bar{V} = V(G) \cup V(G')$. If a node exists only in one graph, it is treated as an isolated padding node in the other graph, and a presence bit is added to its features. The aligned node feature matrices are still denoted by X, X'.

Definition E.2 (Augmented graph drift). Let A, A' be self-looped row-normalized adjacency matrices, let S_q, S'_q be the entry score vectors before soft addressing, and let B, B' be row-normalized entity-to-document anchoring matrices. Define

$$\Delta_X = \|X - X'\|_{2,\infty}, \Delta_A = \|A - A'\|_\infty, \Delta_{\text{seed}} = \left\|S_q - S'_q\right\|_\infty, \Delta_B = \|B - B'\|_\infty, \tag{90}$$

where

$$\|H\|_{2,\infty} = \max_{v\in\bar{V}} \|h_v\|_2, \qquad \|A\|_\infty = \max_v \sum_u |A_{vu}|. \tag{91}$$

For the gate input $z_{uv}^{(l)}$ at layer l, define the weighted structural drift as

$$\Delta_Z^{(l)} = \max_v \sum_u A'_{vu} \left\|z_{uv}^{(l)} - z_{uv}^{\prime(l)}\right\|_2, \qquad \Delta_Z = \max_{1\le l\le L} \Delta_Z^{(l)}. \tag{92}$$

The total augmented graph drift is defined as

$$\Delta_{\text{aug}}(G, G'; q) = \Delta_X + \Delta_A + \Delta_{\text{seed}} + \Delta_Z + \Delta_B. \tag{93}$$

E.2 Stability Assumptions

Assumption E.3 (Normalized adjacency). For all considered graphs, $A_{vu} \ge 0$ and

$$\sum_u A_{vu} = 1, \qquad \forall v \in \bar{V}. \tag{94}$$

Therefore, $\|A\|_\infty = 1$. This assumption allows high-degree hubs to exist, but prevents single-layer propagation from being unboundedly amplified by node degree.

Assumption E.4 (Trajectory-local boundedness). For graph pairs G, G' on the training and inference trajectories, there exist constants B_l such that

$$\left\|H^{(l)}(q,G)\right\|_{2,\infty} \le B_l, \qquad \left\|H^{(l)}(q,G')\right\|_{2,\infty} \le B_l, \qquad l = 0, \dots, L. \tag{95}$$

Assumption E.5 (Locally Lipschitz modules). In a neighborhood of the training trajectory, the l-th layer satisfies

$$\left\|W_m^{(l)}\right\|_2 \le M_l, \tag{96}$$

$$\left\|\mathrm{MLP}_g^{(l)}(z) - \mathrm{MLP}_g^{(l)}(z')\right\|_\infty \le L_{g,l} \|z - z'\|_2. \tag{97}$$

The Lipschitz constant of PReLU is L_σ, and the Lipschitz constant of LayerNorm with a numerical stabilizer in this trajectory neighborhood is $L_{\mathrm{LN},l}$.

Assumption E.6 (Local Lipschitzness of score head and projection). The entity score head satisfies

$$\|s_E(q,G) - s_E(q,G')\|_\infty \le L_E \, \|H(q,G) - H(q,G')\|_{2,\infty}. \tag{98}$$

Meanwhile, $\|s_E(q,G)\|_\infty \le S_E$, and B is row-normalized, so $\|B\|_\infty \le 1$.

E.3 Stability of Soft Addressing and Initial Representation

Lemma E.7 (Softmax and pre-activation stability). *Let*

$$p = \mathrm{softmax}(S/T_0), \qquad p' = \mathrm{softmax}(S'/T_0). \tag{99}$$

Then

$$\|p - p'\|_\infty \le \frac{1}{T_0} \|S - S'\|_\infty. \tag{100}$$

Furthermore, let $a_v = (p_v + \epsilon_p)^\eta$ *and* $a'_v = (p'_v + \epsilon_p)^\eta$, *where* $0 < \eta \le 1$. *Then*

$$\|a - a'\|_\infty \le \frac{\eta \epsilon_p^{\eta-1}}{T_0} \|S - S'\|_\infty. \tag{101}$$

Proof. The Jacobian of softmax is $J(z) = \mathrm{diag}(p) - pp^\top$. For any row i,

$$\sum_j \left|J_{ij}(z)\right| = 2p_i(1-p_i) \le 1. \tag{102}$$

Thus, $\|J(z)\|_{\infty\to\infty} \le 1$. By the mean value theorem and setting $z = S/T_0$, we obtain

$$\|p - p'\|_\infty \le \frac{1}{T_0} \|S - S'\|_\infty. \tag{103}$$

The function $t \mapsto (t+\epsilon_p)^\eta$ is Lipschitz on $[0,1]$, with constant at most $\eta\epsilon_p^{\eta-1}$. Composing the two inequalities gives the conclusion. □

Lemma E.8 (Initial node representation stability). *Let* $u_q = W_q \, \mathrm{Emb}(q)$, *and define*

$$h_v^{(0)} = a_v(q) u_q + W_x x_v. \tag{104}$$

Then

$$\left\| H^{(0)}(q,G) - H^{(0)}(q,G') \right\|_{2,\infty} \le C_{\mathrm{init}} (\Delta_{\mathrm{seed}} + \Delta_X), \tag{105}$$

where

$$C_{\mathrm{init}} = \frac{\eta \epsilon_p^{\eta-1}}{T_0} \left\|u_q\right\|_2 + \|W_x\|_2. \tag{106}$$

Proof. For any node v,

$$h_v^{(0)} - h_v'^{(0)} = (a_v - a'_v) u_q + W_x (x_v - x'_v). \tag{107}$$

Taking the norm and applying Lemma 4.8 gives

$$\left\| h_v^{(0)} - h_v'^{(0)} \right\|_2 \le \frac{\eta \epsilon_p^{\eta-1}}{T_0} \left\|u_q\right\|_2 \Delta_{\mathrm{seed}} + \|W_x\|_2 \Delta_X. \tag{108}$$

Taking the maximum over v gives the conclusion. □

E.4 Single-layer Stability of Structurally Gated Propagation

Lemma E.9 (Boundedness and stability of structural gate). *The structural gate of layer* l,

$$g_{uv}^{(l)} = 1 + \delta \tanh(\mathrm{MLP}_g^{(l)}(z_{uv}^{(l)})), \tag{109}$$

satisfies

$$\left\| g_{uv}^{(l)} \right\|_\infty \le 1 + \delta, \tag{110}$$

and

$$\left\| g_{uv}^{(l)} - g_{uv}'^{(l)} \right\|_\infty \le \delta L_{g,l} \left\| z_{uv}^{(l)} - z_{uv}'^{(l)} \right\|_2. \tag{111}$$

Proof. Since the range of tanh is contained in $[-1,1]$, the first statement follows immediately. Moreover, because tanh is 1-Lipschitz and $\mathrm{MLP}_g^{(l)}$ is $L_{g,l}$-Lipschitz in the trajectory neighborhood,

$$\begin{aligned} \left\| g_{uv}^{(l)} - g_{uv}'^{(l)} \right\|_\infty &\le \delta \left\| \mathrm{MLP}_g^{(l)}(z_{uv}^{(l)}) - \mathrm{MLP}_g^{(l)}(z_{uv}'^{(l)}) \right\|_\infty \\ &\le \delta L_{g,l} \left\| z_{uv}^{(l)} - z_{uv}'^{(l)} \right\|_2. \end{aligned} \tag{112}$$

□

Lemma E.10 (Single-layer stability of structurally gated propagation). *Define*

$$D_l = \left\| H^{(l)}(q,G) - H^{(l)}(q,G') \right\|_{2,\infty}. \tag{113}$$

Under Assumptions 4.5–4.7, the l*-th propagation layer satisfies*

$$D_l \le \alpha_l D_{l-1} + \beta_l^A \Delta_A + \beta_l^Z \Delta_Z^{(l)}, \tag{114}$$

where one can take

$$\alpha_l = L_{\mathrm{LN},l}\big(1 + L_\sigma(1+\delta) M_l\big), \tag{115}$$

$$\beta_l^A = L_{\mathrm{LN},l} L_\sigma (1+\delta) M_l B_{l-1}, \qquad \beta_l^Z = L_{\mathrm{LN},l} L_\sigma \delta L_{g,l} M_l B_{l-1}. \tag{116}$$

Proof. Write

$$M_v = \sum_u A_{vu} g_{uv} \odot W h_u, \qquad M'_v = \sum_u A'_{vu} g'_{uv} \odot W h'_u, \tag{117}$$

where the layer index l is omitted. Adding and subtracting intermediate terms gives

$$\begin{aligned} M_v - M'_v = &\sum_u A_{vu} g_{uv} \odot W(h_u - h'_u) \\ &+ \sum_u (A_{vu} - A'_{vu}) g_{uv} \odot W h'_u \\ &+ \sum_u A'_{vu} (g_{uv} - g'_{uv}) \odot W h'_u. \end{aligned} \tag{118}$$

The first term is controlled by row-normalization, $\|g_{uv}\|_\infty \le 1+\delta$, and $\|W\|_2 \le M_l$:

$$\left\| \sum_u A_{vu} g_{uv} \odot W(h_u - h'_u) \right\|_2 \le (1+\delta) M_l D_{l-1}. \tag{119}$$

The second term satisfies

$$\left\| \sum_u (A_{vu} - A'_{vu}) g_{uv} \odot W h'_u \right\|_2 \le (1+\delta) M_l B_{l-1} \sum_u \left|A_{vu} - A'_{vu}\right| \tag{120}$$

$$\le (1+\delta) M_l B_{l-1} \Delta_A. \tag{121}$$

For the third term, by Lemma 4.10,

$$\left\|\sum_u A'_{vu}(g_{uv}-g'_{uv})\odot Wh'_u\right\|_2 \le \delta L_{g,l}M_lB_{l-1}\sum_u A'_{vu}\left\|z^{(l)}_{uv}-z'^{(l)}_{uv}\right\|_2 \le \delta L_{g,l}M_lB_{l-1}\Delta^{(l)}_Z. \quad (122)$$

Therefore,

$$\left\|M_v-M'_v\right\|_2 \le (1+\delta)M_lD_{l-1}+(1+\delta)M_lB_{l-1}\Delta_A+\delta L_{g,l}M_lB_{l-1}\Delta^{(l)}_Z. \quad (123)$$

By the L_σ-Lipschitz property of PReLU, the residual structure, and the $L_{\mathrm{LN},l}$-Lipschitz property of LayerNorm,

$$\left\|h^{(l)}_v-h'^{(l)}_v\right\|_2 \le L_{\mathrm{LN},l}\left(\left\|h^{(l-1)}_v-h'^{(l-1)}_v\right\|_2+L_\sigma\left\|M_v-M'_v\right\|_2\right). \quad (124)$$

Taking the maximum over v gives Eq. (114). □

E.5 Stability of Representations, Scores, and Retrieval Sets

Theorem E.11 (Local stability of structurally gated representations to augmented graph drift). *Under Assumptions 4.5–4.7, L-layer structurally gated propagation satisfies*

$$D_L \le \left(\prod_{l=1}^{L}\alpha_l\right)D_0+\sum_{t=1}^{L}\left(\prod_{l=t+1}^{L}\alpha_l\right)\left(\beta^A_t\Delta_A+\beta^Z_t\Delta^{(t)}_Z\right). \quad (125)$$

Therefore, there exists a constant $C_H>0$ such that

$$\left\|H^{(L)}(q,G)-H^{(L)}(q,G')\right\|_{2,\infty} \le C_H(\Delta_X+\Delta_{\mathrm{seed}}+\Delta_A+\Delta_Z). \quad (126)$$

Proof. By Lemma 4.11, the recursion in Eq. (114) holds. Unrolling the recursion layer by layer gives Eq. (125). By Lemma 4.9,

$$D_0 \le C_{\mathrm{init}}(\Delta_X+\Delta_{\mathrm{seed}}). \quad (127)$$

Substituting $\Delta^{(t)}_Z \le \Delta_Z$ and merging all layer-related constants into C_H gives the conclusion. □

Theorem E.12 (Stability of context/schema dual channels). *If the two channels respectively satisfy*

$$\|H_{\mathrm{ctx}}(q,G)-H_{\mathrm{ctx}}(q,G')\|_{2,\infty} \le C_{\mathrm{ctx}}\Delta_{\mathrm{aug}}, \quad (128)$$

$$\|H_{\mathrm{sch}}(q,G)-H_{\mathrm{sch}}(q,G')\|_{2,\infty} \le C_{\mathrm{sch}}\Delta_{\mathrm{aug}}, \quad (129)$$

then the additive fusion in Eq. (??) satisfies

$$\|H(q,G)-H(q,G')\|_{2,\infty} \le (C_{\mathrm{ctx}}+|\beta_{\mathrm{sch}}|\,C_{\mathrm{sch}})\Delta_{\mathrm{aug}}. \quad (130)$$

If a normalized or gated convex fusion theoretical form is adopted,

$$H_\lambda(q,G)=(1-\lambda)H_{\mathrm{ctx}}(q,G)+\lambda H_{\mathrm{sch}}(q,G), \qquad 0\le\lambda\le 1, \quad (131)$$

then

$$\|H_\lambda(q,G)-H_\lambda(q,G')\|_{2,\infty} \le \big((1-\lambda)C_{\mathrm{ctx}}+\lambda C_{\mathrm{sch}}\big)\Delta_{\mathrm{aug}}. \quad (132)$$

In particular, if $C_{\mathrm{sch}}<C_{\mathrm{ctx}}$, increasing λ decreases this worst-case stability upper bound.

Proof. The additive fusion case follows directly from the triangle inequality:

$$\|H(G)-H(G')\|_{2,\infty} \le \|H_{\mathrm{ctx}}(G)-H_{\mathrm{ctx}}(G')\|_{2,\infty} + |\beta_{\mathrm{sch}}|\,\|H_{\mathrm{sch}}(G)-H_{\mathrm{sch}}(G')\|_{2,\infty} \le (C_{\mathrm{ctx}}+|\beta_{\mathrm{sch}}|\,C_{\mathrm{sch}})\Delta_{\mathrm{aug}}. \quad (133)\text{–}(134)$$

The convex fusion case is analogous:

$$\|H_\lambda(G)-H_\lambda(G')\|_{2,\infty} \le (1-\lambda)C_{\mathrm{ctx}}\Delta_{\mathrm{aug}}+\lambda C_{\mathrm{sch}}\Delta_{\mathrm{aug}}. \quad (135)$$

If $C_{\mathrm{sch}}<C_{\mathrm{ctx}}$, the right-hand side is monotonically decreasing in λ. □

Remark E.13. Equation (130) does not claim that the schema prior necessarily reduces the worst-case Lipschitz constant of additive fusion; its main theoretical role also lies in risk decomposition and sample complexity reduction. If additional gating, normalization, or regularization constraints are adopted in the implementation, then Eq. (132) shows that the schema channel can also serve as a low-sensitivity channel to reduce graph evolution drift.

Theorem E.14 (Stability of entity scores and document scores). *Under Assumption 4.7, there exist constants $C_E, C_D>0$ such that*

$$\|s_E(q,G)-s_E(q,G')\|_\infty \le C_E\Delta_{\mathrm{aug}}, \quad (136)$$

$$\|s_D(q,G)-s_D(q,G')\|_\infty \le C_D\Delta_{\mathrm{aug}}. \quad (137)$$

For additive fusion, one can take

$$C_E=L_E(C_{\mathrm{ctx}}+|\beta_{\mathrm{sch}}|\,C_{\mathrm{sch}}), \qquad C_D=C_E+S_E. \quad (138)$$

Proof. The entity score upper bound follows from the Lipschitz property of the score head and Theorem 4.13. For document projection, $s_D=Bs_E$, and therefore

$$s_D(G)-s_D(G')=B_G(s_E(G)-s_E(G'))+(B_G-B_{G'})s_E(G'). \quad (139)$$

Taking the ℓ_∞ norm and using $\|B_G\|_\infty\le 1$ gives

$$\|s_D(G)-s_D(G')\|_\infty \le \|s_E(G)-s_E(G')\|_\infty+\|B_G-B_{G'}\|_\infty\,\|s_E(G')\|_\infty. \quad (140)$$

Using $\|s_E(G')\|_\infty\le S_E$ and $\Delta_B\le\Delta_{\mathrm{aug}}$ gives the document score stability. □

Theorem E.15 (Boundary stability of hard top-k). *Let $s=s_D(q,G)$ and $s'=s_D(q,G')$, and suppose*

$$\|s-s'\|_\infty \le \epsilon_s. \quad (141)$$

Let $t_k=s_{(k)}$ be the k-th largest score in s, and define the boundary set

$$\mathcal{B}_{k,2\epsilon_s}(s)=\{d:|s_d-t_k|\le 2\epsilon_s\}. \quad (142)$$

Then

$$\text{Top-}k(s)\triangle\text{Top-}k(s') \subseteq \mathcal{B}_{k,2\epsilon_s}(s), \quad (143)$$

and hence

$$|\text{Top-}k(s)\triangle\text{Top-}k(s')| \le \left|\mathcal{B}_{k,2\epsilon_s}(s)\right|. \quad (144)$$

In particular, if $s_{(k)}-s_{(k+1)}>2\epsilon_s$, then Top-$k(s)$ = Top-$k(s')$.

Proof. Take any $i\in\text{Top-}k(s)\setminus\text{Top-}k(s')$. Since i drops out of the top-k, there exists $j\notin\text{Top-}k(s)$ such that $j\in\text{Top-}k(s')$ and $s'_j\ge s'_i$. By the perturbation bound,

$$s_j+\epsilon_s\ge s'_j\ge s'_i\ge s_i-\epsilon_s, \quad (145)$$

so $s_i\le s_j+2\epsilon_s\le t_k+2\epsilon_s$. Since $i\in\text{Top-}k(s)$, we have $s_i\ge t_k$, and hence $i\in\mathcal{B}_{k,2\epsilon_s}(s)$. A symmetric argument for $j\in\text{Top-}k(s')\setminus\text{Top-}k(s)$ gives $j\in\mathcal{B}_{k,2\epsilon_s}(s)$. Therefore, the symmetric difference is contained in the boundary set. If $s_{(k)}-s_{(k+1)}>2\epsilon_s$, boundary exchange cannot occur, and the top-k set remains unchanged. □

Corollary E.16 (Top-k boundary stability under graph evolution). *By Theorem 4.15, taking $\epsilon_s = C_D\Delta_{\text{aug}}(G, G'; q)$ yields*

$$P_k(q, G) \triangle P_k(q, G') \subseteq \mathcal{B}_{k, 2C_D\Delta_{\text{aug}}}(s_D(q, G)). \tag{146}$$

Therefore, the instability of hard top-k is restricted to candidates near the original score boundary.

Theorem E.17 (Stability of soft retrieval distribution). *Let*

$$\pi_D(q, G) = \text{softmax}(s_D(q, G)/\tau). \tag{147}$$

If $\|s_D(q, G) - s_D(q, G')\|_\infty \le \epsilon_s$, then

$$\|\pi_D(q, G) - \pi_D(q, G')\|_1 \le \frac{2}{\tau}\epsilon_s. \tag{148}$$

Therefore,

$$\|\pi_D(q, G) - \pi_D(q, G')\|_1 \le \frac{2C_D}{\tau}\Delta_{\text{aug}}(G, G'; q). \tag{149}$$

Proof. The Jacobian of softmax is $J(z) = \text{diag}(\pi) - \pi\pi^\top$. For any perturbation r,

$$J(z)r = \pi \odot (r - \mathbb{E}_\pi r). \tag{150}$$

If $\|r\|_\infty \le 1$, then $|r_i - \mathbb{E}_\pi r| \le 2$, so

$$\|J(z)r\|_1 \le \sum_i \pi_i |r_i - \mathbb{E}_\pi r| \le 2. \tag{151}$$

Thus, the $\ell_\infty \to \ell_1$ Lipschitz constant of softmax is at most 2. Since the input is s_D/τ, we obtain

$$\|\pi_D(s) - \pi_D(s')\|_1 \le \frac{2}{\tau}\|s - s'\|_\infty. \tag{152}$$

Substituting Theorem 4.15 gives the conclusion. □

Theorem E.18 (High-probability graph evolution stability). *If the writer's single-round graph update satisfies*

$$\mathbb{P}[\Delta_{\text{aug}}(G, G'; q) > \epsilon] \le \delta, \tag{153}$$

then

$$\mathbb{P}\left[\|s_D(q, G) - s_D(q, G')\|_\infty > C_D\epsilon\right] \le \delta. \tag{154}$$

If $\mathbb{E}[\Delta_{\text{aug}}(G, G'; q)] \le \bar{\epsilon}$, then

$$\mathbb{E}\left[\|s_D(q, G) - s_D(q, G')\|_\infty\right] \le C_D\bar{\epsilon}. \tag{155}$$

Proof. By Theorem 4.15, for any graph pair, we have

$$\|s_D(q, G) - s_D(q, G')\|_\infty \le C_D\Delta_{\text{aug}}(G, G'; q). \tag{156}$$

Therefore, the event $\{\|s_D(q, G) - s_D(q, G')\|_\infty > C_D\epsilon\}$ implies the event $\{\Delta_{\text{aug}}(G, G'; q) > \epsilon\}$, so the probability upper bound follows immediately. The expectation conclusion follows by taking expectations on both sides of the deterministic inequality. □

E.6 Local Influence Cone

Proposition 8 (Influence cone of local graph updates). Suppose the writer only changes nodes, edges, anchors, or attributes on a primitive set $\mathcal{U}$. Suppose that the structural gate input $z_{uv}^{(l)}$, except for the graph-level summary, only depends on a local neighborhood of radius r_z, and that G and G' are exactly the same outside $\mathcal{N}_{L+r_z}(\mathcal{U})$. If graph-level summary drift is ignored, then for any

$$v \notin \mathcal{N}_{L+r_z}(\mathcal{U}), \tag{157}$$

we have

$$h_v^{(L)}(q, G) = h_v^{(L)}(q, G'). \tag{158}$$

If the graph-level summary drift is $\rho_g = \|r_G - r_{G'}\|_2$, then there exists a constant C_g such that

$$\left\|h_v^{(L)}(q, G) - h_v^{(L)}(q, G')\right\|_2 \le C_g\rho_g. \tag{159}$$

Proof. First consider the case without graph-level summary drift. We induct on the layer index l. For $l = 0$, if $v \notin \mathcal{N}_{L+r_z}(\mathcal{U})$, then its node features, presence bit, and seed score are all identical, so $h_v^{(0)}(G) = h_v^{(0)}(G')$. Suppose that at layer $l-1$, all nodes whose distance from $\mathcal{U}$ exceeds $L + r_z - (l-1)$ have identical representations. If $v \notin \mathcal{N}_{L+r_z-l}(\mathcal{U})$, then all its one-hop neighbors u do not belong to $\mathcal{N}_{L+r_z-(l-1)}(\mathcal{U})$; by the induction hypothesis, $h_u^{(l-1)}(G) = h_u^{(l-1)}(G')$. Meanwhile, the radius-$r_z$ local structural contexts of all relevant edges are also identical, so the gate, message multiset, and aggregation result are identical, and hence $h_v^{(l)}(G) = h_v^{(l)}(G')$. Taking $l = L$ gives the first conclusion. If graph-level summary drift exists, then the gate input has an additional uniform perturbation term ρ_g, and a $C_g\rho_g$-type upper bound follows from Lemma 4.10 and the recursion in Theorem 4.12. □

F Theoretical Motivation of the Self-evolving Writer–Reader Loop

F.1 Joint Memory Utility

The reader-aware writer reward can consist of evidence coverage, precision, deducibility, and answer utility. Abstractly, define the joint memory utility as

$$\mathcal{J}(\theta, \phi) = \mathbb{E}_{(q,D,D^+,y)}\left[U\left(R_\phi(q, W_\theta(q, D), D), D^+, y\right)\right], \tag{160}$$

where U can be taken as

$$U = \frac{\alpha r_{\text{rec}} + \beta r_{\text{pre}} + \gamma r_{\text{ded}}}{\alpha + \beta + \gamma} - \lambda_{\text{rep}}\rho_{\text{rep}} + \lambda_{\text{fmt}} r_{\text{fmt}}, \tag{161}$$

or an extended form including answer-level reward. This definition places the writer's graph construction quality and the reader's graph reading ability under the same objective.

F.2 Approximate Coordinate Improvement

Theorem F.1 (The self-evolution process is approximate coordinate improvement on joint utility). *Suppose that the writer update at round r satisfies*

$$\mathcal{J}(\theta^{(r+1)}, \phi^{(r)}) \ge \mathcal{J}(\theta^{(r)}, \phi^{(r)}) + \Delta_W^{(r)} - \epsilon_W^{(r)}, \tag{162}$$

and the reader update satisfies

$$\mathcal{J}(\theta^{(r+1)}, \phi^{(r+1)}) \ge \mathcal{J}(\theta^{(r+1)}, \phi^{(r)}) + \Delta_R^{(r)} - \epsilon_R^{(r)}. \tag{163}$$

Then one full round of writer–reader self-evolution satisfies

$$\mathcal{J}(\theta^{(r+1)},\phi^{(r+1)})-\mathcal{J}(\theta^{(r)},\phi^{(r)})\geq\Delta_W^{(r)}+\Delta_R^{(r)}-\epsilon_W^{(r)}-\epsilon_R^{(r)}. \quad (164)$$

Therefore, as long as $\Delta_W^{(r)}+\Delta_R^{(r)}>\epsilon_W^{(r)}+\epsilon_R^{(r)}$, the joint memory utility improves in that round.

Proof. By telescoping decomposition,

$$\begin{aligned}&\mathcal{J}(\theta^{(r+1)},\phi^{(r+1)})-\mathcal{J}(\theta^{(r)},\phi^{(r)})\\&=\left[\mathcal{J}(\theta^{(r+1)},\phi^{(r+1)})-\mathcal{J}(\theta^{(r+1)},\phi^{(r)})\right]+\left[\mathcal{J}(\theta^{(r+1)},\phi^{(r)})-\mathcal{J}(\theta^{(r)},\phi^{(r)})\right]\end{aligned} \quad (165)$$

Substituting Eq. (162) and Eq. (163), respectively, gives the conclusion. □

F.3 Reader Reward Bias and Calibration Benefit

Definition F.2 (True utility and reader surrogate reward)**.** Let $U^\star(G)$ denote the true utility of graph memory G with respect to the downstream task, and let $\widehat{U}_\phi(G)$ denote the surrogate reward constructed from the readout result of reader R_ϕ. We say that the reader reward bias is at most ϵ_ϕ if, for all considered graphs G,

$$\left|\widehat{U}_\phi(G)-U^\star(G)\right|\leq\epsilon_\phi. \quad (166)$$

Theorem F.3 (Surrogate reward improvement to true utility improvement)**.** *If the reader reward bias is at most ϵ_ϕ, and the writer update improves the surrogate reward by*

$$\widehat{U}_\phi(G_{\theta'})-\widehat{U}_\phi(G_\theta)\geq\Delta, \quad (167)$$

then the true utility satisfies

$$U^\star(G_{\theta'})-U^\star(G_\theta)\geq\Delta-2\epsilon_\phi. \quad (168)$$

Proof. By the bias assumption,

$$U^\star(G_{\theta'})\geq\widehat{U}_\phi(G_{\theta'})-\epsilon_\phi,\qquad U^\star(G_\theta)\leq\widehat{U}_\phi(G_\theta)+\epsilon_\phi. \quad (169)$$

Subtracting the two inequalities gives

$$U^\star(G_{\theta'})-U^\star(G_\theta)\geq\widehat{U}_\phi(G_{\theta'})-\widehat{U}_\phi(G_\theta)-2\epsilon_\phi\geq\Delta-2\epsilon_\phi. \quad (170)$$

□

Corollary F.4 (Reader calibration reduces writer optimization bias)**.** *If the reader is calibrated from ϕ to ϕ' and reduces the reward bias from ϵ_ϕ to $\epsilon_{\phi'}$, where $\epsilon_{\phi'}<\epsilon_\phi$, then for the same surrogate reward improvement Δ, the lower bound on true utility improvement increases by*

$$2(\epsilon_\phi-\epsilon_{\phi'}). \quad (171)$$

Proof. By Theorem 5.3, the true utility improvement lower bound before calibration is $\Delta-2\epsilon_\phi$, and after calibration it is $\Delta-2\epsilon_{\phi'}$. Subtracting the two gives the result. □

F.4 Irreducible Bottlenecks of Single-sided Updates

Proposition 9 (Lower-bound bottlenecks of single-sided updates). Assume that the overall error can be decomposed as

$$\mathcal{E}(\theta,\phi)=\mathcal{E}_{\text{write}}(\theta)+\mathcal{E}_{\text{read}}(\phi;\theta)+\epsilon_{\text{int}}(\theta,\phi), \quad (172)$$

where all terms are nonnegative. If only the reader is updated, i.e., $\phi\mapsto\phi'$ while θ is fixed, then

$$\mathcal{E}(\theta,\phi')\geq\mathcal{E}_{\text{write}}(\theta). \quad (173)$$

If only the writer is updated, i.e., $\theta\mapsto\theta'$ while ϕ is fixed, then

$$\mathcal{E}(\theta',\phi)\geq\mathcal{E}_{\text{read}}(\phi;\theta'). \quad (174)$$

Therefore, reader-only updates cannot compensate for evidence chains that the writer has not written; writer-only updates cannot guarantee that a fixed reader can read out the evidence structures in the new graph distribution.

Proof. By the decomposition in Eq. (172) and the nonnegativity of all terms,

$$\mathcal{E}(\theta,\phi')=\mathcal{E}_{\text{write}}(\theta)+\mathcal{E}_{\text{read}}(\phi';\theta)+\epsilon_{\text{int}}(\theta,\phi')\geq\mathcal{E}_{\text{write}}(\theta). \quad (175)$$

The second inequality is analogous. □

F.5 Stability of Closed-loop Graph Evolution and Parameter Updates

Theorem F.5 (Score drift control under multi-round self-evolution)**.** *Let the graph at round r be $G^{(r)}$, and the reader parameter be $\phi^{(r)}$. If single-step graph stability satisfies*

$$\left\|s_D(q,G^{(r+1)};\phi^{(r)})-s_D(q,G^{(r)};\phi^{(r)})\right\|_\infty\leq C_D\Delta_r, \quad (176)$$

where $\Delta_r=\Delta_{\text{aug}}(G^{(r)},G^{(r+1)};q)$; and if the score is locally Lipschitz with respect to the parameter:

$$\|s_D(q,G;\phi)-s_D(q,G;\phi')\|_\infty\leq C_\phi\,\|\phi-\phi'\|_2, \quad (177)$$

then

$$\begin{aligned}&\left\|s_D(q,G^{(T)};\phi^{(T)})-s_D(q,G^{(0)};\phi^{(0)})\right\|_\infty\\&\leq\sum_{r=0}^{T-1}\left(C_D\Delta_r+C_\phi\left\|\phi^{(r+1)}-\phi^{(r)}\right\|_2\right).\end{aligned} \quad (178)$$

Proof. For each r, adding and subtracting the intermediate term $s_D(q,G^{(r+1)};\phi^{(r)})$ gives

$$\begin{aligned}&\left\|s_D(q,G^{(r+1)};\phi^{(r+1)})-s_D(q,G^{(r)};\phi^{(r)})\right\|_\infty\\&\leq\left\|s_D(q,G^{(r+1)};\phi^{(r+1)})-s_D(q,G^{(r+1)};\phi^{(r)})\right\|_\infty \quad (179)\\&+\left\|s_D(q,G^{(r+1)};\phi^{(r)})-s_D(q,G^{(r)};\phi^{(r)})\right\|_\infty\\&\leq C_\phi\left\|\phi^{(r+1)}-\phi^{(r)}\right\|_2+C_D\Delta_r. \quad (180)\end{aligned}$$

Summing over $r=0,\dots,T-1$ and using the triangle inequality gives the conclusion. □

Corollary F.6 (High-probability multi-round stability). *If* $\mathbb{P}[\Delta_r > \epsilon_r] \le \delta_r$, *then with probability at least* $1 - \sum_{r=0}^{T-1} \delta_r$,

$$\left\| s_D(q, G^{(T)}; \phi^{(T)}) - s_D(q, G^{(0)}; \phi^{(0)}) \right\|_\infty \le \sum_{r=0}^{T-1} \left(C_D \epsilon_r + C_\phi \left\| \phi^{(r+1)} - \phi^{(r)} \right\|_2 \right). \tag{181}$$

Proof. By a union bound, the event $\{\forall r, \Delta_r \le \epsilon_r\}$ holds with probability at least $1 - \sum_r \delta_r$. Applying Theorem 5.7 on this event gives the conclusion. □

G Analysis of the Memory Writer

This section further analyzes the memory writer while keeping the memory reader fixed. The experiments mainly use HotpotQA and MuSiQue. To further examine domain transfer capability, we also evaluate the trained writing policy on GRBench-Amazon, HaluMem-Medium, and LongMemEval-Oracle [**? ?**]. We primarily report Precision, Recall, and Deducible: the first two measure whether the text retrieved by the reader covers the gold supporting contexts, while Deducible is determined by a judge as to whether the standard answer can be inferred from the retrieved context, thus more directly reflecting whether the graph memory is usable for reasoning.

Table 5 shows that different rewards have different preferences. Overall, RL-Hybrid achieves the best overall results, indicating that hybrid rewards can simultaneously constrain the selectivity and coverage of graph writing. *Hybrid + frozen answer API* achieves the highest Deducible but slightly lower retrieval Precision/Recall, suggesting that answer-side feedback helps improve reasoning usability, but may also make the writer more conservatively inclined to write evidence that directly supports the answer. Table 6 shows that a writer learned on HotpotQA/MuSiQue can transfer to GRBench, HaluMem, and LongMemEval, but continued training on the target domain still brings significant improvements. This indicates that the memory structure in agent memory scenarios is not entirely similar to that in traditional multi-hop QA, so target-domain feedback remains crucial. Table 7 further shows that the writing protocol and interaction budget affect the trade-off between coverage and noise. Relaxing the tight prompt can improve Recall, but reduces Deducible; increasing iterative turns helps complete cross-document bridging paths, but when the budget is too large or the protocol is too loose. More detailed reader-side sensitivity, training stability, and regularization analyses are provided in Appendix A.

H Ablation Study of the Memory Reader

We conduct ablation studies to isolate the contribution of each major component in the memory reader. All variants use the same writer-produced graph memory and the same retrieval budget unless the ablated component directly changes the retrieval mechanism. The ablations are organized around four questions: (1) whether structured query planning and global soft addressing are necessary for recovering evidence from fragmented cues; (2) whether structurally gated propagation improves over uniform graph propagation; (3) whether cross-graph structural priors and target-graph calibration are both needed for evolving graph memory; and (4) whether reader training and entity-to-document projection are important for converting entity-level activation into document-level retrieval.

Table 8 summarizes the results. The first group evaluates how the reader handles fragmented cues. Removing structured query planning, alias/constraint cues, or global soft addressing forces the reader to rely more heavily on surface-level query matches or a small number of anchor entities, directly testing whether the complete evidence chain can still be recovered when distant bridge nodes are not initially activated. The second group studies whether the reader uses graph structure in a learned and selective way. Removing structural gates or replacing them with uniform message passing tests whether treating hub edges, bridge edges, redundant edges, and noisy shortcuts similarly harms retrieval. The third group examines the context–schema decomposition: the schema channel captures transferable structural reading patterns, while the context channel adapts the reader to the current writer-produced graph. The last group evaluates the selector, entity-to-document projection, GFM pre-training, and supervised retrieval fine-tuning.

The ablation results show the relative contribution of each memory-reader component. First, structured query planning and global soft addressing are important for fragmented-cue retrieval: removing them noticeably weakens performance, especially on MuSiQue and 2WikiMultiHopQA, where evidence chains are more likely to depend on implicit bridge entities. Second, structurally gated propagation consistently improves over uniform message passing, indicating that graph structure should be used selectively rather than as a fixed expansion rule. Among the structural inputs, graph-level summaries have a relatively smaller effect, while node-level and edge-pair features are more important for recognizing hubs, bridges, and cross-community evidence paths. Third, both the schema prior and context calibration channels contribute to performance, suggesting that the reader benefits from preserving transferable structural priors while adapting to the current writer-produced graph. Finally, supervised retrieval fine-tuning is essential for aligning the GFM reader with document-level evidence retrieval, whereas GFM pre-training provides transferable structural initialization that improves stability across datasets.

I Implementation Details of the Query-conditioned Subgraph Selection Regularizer

We provide the implementation details of the query-conditioned subgraph selector. In addition to the base entity scoring, it further learns a soft gating probability $\pi_e(q)$, which characterizes whether entity e should enter the reading subgraph of the current query q. This module performs lightweight reweighting of the final entity score, and constrains the reading subgraph through several structural regularizers during training.

Query-conditioned Selection Probability. Given a query q and a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $\mathbf{h}_e \in \mathbb{R}^d$ denote the representation of entity node $e \in \mathcal{V}$ after propagation by the GFM backbone, and let $\mathbf{z}_q \in \mathbb{R}^d$ denote the query representation output by the query encoder and then linearly projected. Here, d is the hidden dimension. The selector first projects the node representation and the query representation into the same selector space:

$$\mathbf{u}_e = W_n \mathbf{h}_e, \qquad \mathbf{v}_q = W_s \mathbf{z}_q, \tag{182}$$

Table 5: Training results for the memory writer. *GFM-pretrained-only* refers to using rewards fed back only by the pretrained memory reader, while *GFM-finetuned* further refers to using the fine-tuned memory reader.

Methods	Prec.↑	Recall↑	Deducible↑
GFM-pretrained-only	0.838	0.818	0.510
GFM-finetuned	0.824	0.813	0.512
RL-Recall	0.889	0.835	0.502
RL-F1	0.839	0.881	0.497
RL-Deduce	0.861	0.892	0.517
RL-Hybrid	**0.902**	**0.917**	0.522
Hybrid + frozen answer API	0.832	0.874	**0.526**

Table 6: Cross-dataset memory writing results. "Base→Target" indicates direct evaluation on the target domain after training on the HotpotQA/MuSiQue base; "Target train→val" indicates training and validation on the target domain.

Settings	Prec.↑	Recall↑	Deducible↑
Base→GRBench	0.575	0.609	0.411
GRBench train→val	0.794	0.833	0.596
Base→HaluMem	0.230	0.448	0.299
HaluMem train→val	0.312	0.708	0.438
Base→LongMemEval	0.232	0.376	0.475
LongMem train→LongMemEval	0.377	0.439	0.531

Table 7: Ablation of writing protocols and interaction budgets. Tight=True indicates that the writer performs a single-round graph write under a stricter evidence budget, meaning the reader exposes only fewer, higher-confidence candidate pieces of evidence to the writer; Tight=False indicates that this evidence budget is relaxed, allowing the writer to access a broader candidate context. Iterative indicates that multi-round interactive writer–reader writing is enabled: the writer first writes the initial graph memory, the reader then returns retrieval feedback based on the current graph, and the writer continues to supplement or revise the graph structure. Here, 12/20/24 turns indicates the maximum number of interaction rounds allowed, and tight/loose indicates that strict or relaxed evidence budget constraints are still used during this multi-round interaction process.

Settings	Prec.↑	Recall↑	Deducible↑
Tight=True	0.836	0.806	0.515
Tight=False	0.845	0.851	0.506
Iterative, 12 turns, tight	0.852	0.829	0.516
Iterative, 20 turns, tight	0.835	**0.881**	0.522
Iterative, 24 turns, loose	**0.863**	0.826	**0.531**

where $W_n \in \mathbb{R}^{d_s \times d}$ is the node-side projection matrix, $W_s \in \mathbb{R}^{d_s \times d}$ is the query-side projection matrix, and d_s is the hidden dimension of the selector space. In our implementation, we set $d_s = d$, but the two do not have to be equal. Then, the selector obtains the selection logit of entity e with respect to query q through a scaled inner product:

$$\zeta_e(q) = \frac{\mathbf{u}_e^\top \mathbf{v}_q}{T_s}, \tag{183}$$

where $T_s > 0$ is the selector temperature coefficient. The final soft selection probability is defined as

$$\pi_e(q) = \mathrm{sigmoid}(\zeta_e(q)) = \frac{1}{1 + \exp(-\zeta_e(q))}. \tag{184}$$

Here, $\pi_e(q) \in (0, 1)$ can be understood as the soft probability that entity e is included in the reading subgraph of the current query. During training, we directly use $\pi_e(q)$ for differentiable optimization; during inference, we can either continue to use the soft probability for reweighting, or obtain a discrete subgraph according to a threshold τ_π:

$$\mathcal{V}_q = \{e \in \mathcal{V} \mid \pi_e(q) > \tau_\pi\}, \qquad \mathcal{E}_q = \{(u, v) \in \mathcal{E} \mid u, v \in \mathcal{V}_q\}. \tag{185}$$

Let $a_e(q)$ denote the base entity score given by the GFM backbone reader. This score is usually obtained from the similarity between the node representation and the query representation, for example

$$a_e(q) = \mathbf{h}_e^\top \mathbf{z}_q. \tag{186}$$

Finally, we have:

$$a_e^{\mathrm{final}}(q) = a_e(q) + \lambda_s \zeta_e(q), \tag{187}$$

Table 8: Ablation study of the memory reader on multi-hop QA retrieval. We report document-level Recall (%) at top-2 and top-5. All variants use the same writer-produced graph memory unless otherwise specified.

Reader Variant	HotpotQA		MuSiQue		2WikiMultiHopQA	
	R@2	R@5	R@2	R@5	R@2	R@5
SAGE	**65.1**	**77.6**	**43.2**	**53.1**	**83.6**	**88.6**
Query planning and global addressing						
SAGE w/o Structured Query Planning	62.7	75.1	40.4	50.1	80.7	86.6
SAGE w/o Global Soft Addressing	59.3	72.5	37.6	47.4	75.9	83.1
SAGE w/o Alias and Constraint Cues	63.0	75.8	41.0	50.8	80.8	86.9
SAGE w/ Anchor-only Initialization	58.6	71.4	36.8	46.5	74.2	82.4
Structurally conditioned propagation						
SAGE w/o Structural Gate	60.4	73.2	39.2	48.7	78.1	84.9
SAGE w/o Node Structural Features	62.1	74.8	40.5	50.0	80.0	86.0
SAGE w/o Edge-pair Structural Features	61.5	74.0	40.1	49.4	79.1	85.6
SAGE w/o Graph-level Summary	63.2	75.9	41.6	51.0	81.3	87.0
SAGE w/ Uniform Message Passing	58.9	71.8	37.5	46.9	75.3	82.9
Cross-graph priors and target-graph calibration						
SAGE w/o Schema Prior Channel	62.4	75.0	40.9	50.6	80.4	86.4
SAGE w/o Context Calibration Channel	61.8	74.3	40.0	49.8	79.5	85.9
SAGE w/o Context–Schema Fusion	60.7	73.5	39.1	48.6	77.8	84.7
Selector, projection, and reader training						
SAGE w/o Controlled Entity-to-Document Projection	60.9	73.9	38.7	48.2	77.2	84.4
SAGE w/o Query-conditioned Selector	63.9	76.5	41.9	52.0	82.2	87.6
SAGE w/ Vanilla GNN Reader	57.2	70.6	36.3	45.2	72.8	80.7

where $\lambda_s \geq 0$ controls the influence of the selector logit on the final entity ranking.

Query–Subgraph Contrastive Regularizer. Using only Eq. (187) to fuse the selector score can easily lead to two types of degeneration: first, the selector may assign high probabilities to most nodes, thereby degenerating into full-graph activation; second, the selector may only learn local high-frequency entities, without forming a subgraph representation that is consistent with the overall semantics of the query. To this end, we first construct a query-conditioned subgraph representation weighted by the selection probabilities:

$$\bar{\mathbf{h}}_\pi(q) = \frac{\sum_{e\in\mathcal{V}} \pi_e(q)\mathbf{h}_e}{\sum_{e\in\mathcal{V}} \pi_e(q) + \epsilon}, \tag{188}$$

where $\epsilon > 0$ is a numerical stability term, which avoids an excessively small denominator when all $\pi_e(q)$ are close to 0. $\bar{\mathbf{h}}_\pi(q)$ can be understood as the semantic center of the soft subgraph activated by the current selector.

For a mini-batch $\mathcal{B} = \{q_i\}_{i=1}^B$, we treat $(\bar{\mathbf{h}}_\pi(q_i), \mathbf{z}_{q_i})$ within the same sample as a positive pair, and treat $(\bar{\mathbf{h}}_\pi(q_i), \mathbf{z}_{q_j})$, $j \neq i$, as in-batch negative pairs. The query–subgraph contrastive loss is defined as

$$\Omega_{\text{nce}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp\left(\text{sim}(\bar{\mathbf{h}}_\pi(q_i), \mathbf{z}_{q_i})/T_n\right)}{\sum_{j=1}^{B} \exp\left(\text{sim}(\bar{\mathbf{h}}_\pi(q_i), \mathbf{z}_{q_j})/T_n\right)}, \tag{189}$$

where $T_n > 0$ is the contrastive learning temperature coefficient, and $\text{sim}(\cdot,\cdot)$ is the similarity function. In implementation, we usually apply ℓ_2 normalization to $\bar{\mathbf{h}}_\pi(q)$ and $\mathbf{z}_q$, and use inner-product similarity, so that $\text{sim}(\bar{\mathbf{h}}_\pi(q), \mathbf{z}_q) = \frac{\bar{\mathbf{h}}_\pi(q)^\top \mathbf{z}_q}{\|\bar{\mathbf{h}}_\pi(q)\|_2 \|\mathbf{z}_q\|_2}$. This term encourages the soft subgraph activated by the selector to semantically represent the current query, rather than only selecting nodes with high frequency or high centrality in the graph.

Size Regularizer. To prevent the selector from improving recall by activating a large number of nodes, we use the average selection probability as a size penalty:

$$\Omega_{\text{size}} = \frac{1}{|\mathcal{V}|}\sum_{e\in\mathcal{V}} \pi_e(q). \tag{190}$$

This term approximately represents the expected proportion of activated nodes. Minimizing Ω_{size} pushes the model to select a smaller reading subgraph. However, this term cannot be used alone; otherwise, the selector may degenerate into selecting too few nodes

or even no nodes. Therefore, it needs to be jointly optimized with Ω_{nce} and the main retrieval loss: the former ensures query relevance, while the latter ensures that the selected structure can still support correct entity and document recall.

Connectivity Smoothing Regularizer. In addition to controllable size, an effective reading subgraph should also have local structural coherence. If the selection probabilities of adjacent nodes differ too much, the model may form several isolated activated points, making multi-hop paths difficult to explicitly utilize. To this end, we use a smoothing penalty on edges:

$$\Omega_{\text{con}} = \frac{1}{|\mathcal{E}|} \sum_{(u,v)\in\mathcal{E}} (\pi_u(q) - \pi_v(q))^2 . \tag{191}$$

Here, (u, v) is a directed or undirected edge in the graph, depending on whether edge directions are preserved during graph construction. If an undirected graph is used, $\mathcal{E}$ can be viewed as the symmetrized edge set. This term does not force all selected nodes to be strictly connected, but encourages adjacent nodes to have similar selection probabilities. In matrix form, if $\boldsymbol{\pi}(q) \in \mathbb{R}^{|\mathcal{V}|}$ is the selection probability vector composed of $\pi_e(q)$, and $\mathbf{L}$ is the graph Laplacian matrix, then this term is equivalent to Laplacian smoothing:

$$\Omega_{\text{con}} \propto \boldsymbol{\pi}(q)^\top \mathbf{L}\boldsymbol{\pi}(q). \tag{192}$$

Therefore, it is consistent with the classical assumption of graph signal smoothing: query relevance, as a soft signal on the graph, should maintain a certain degree of continuity within local neighborhoods.

Computational Complexity of the Selector Itself. Let the batch size be B, the number of nodes be $n = |\mathcal{V}|$, the number of edges be $m = |\mathcal{E}|$, and the hidden dimension be d. The cost of computing $W_n\mathbf{h}_e$ is $O(Bnd^2)$, the cost of computing $W_s\mathbf{z}_q$ is $O(Bd^2)$, and the cost of the inner-product logits is $O(Bnd)$. If we cache $W_n\mathbf{h}_e$ in advance, this term can be reduced to $O(Bnd)$. For the contrastive term, the cost of the subgraph pooling in Eq. (188) is $O(Bnd)$, and the cost of the in-batch NCE similarity matrix is $O(B^2d)$; the size term has cost $O(Bn)$; the connectivity term needs to traverse edges and has cost $O(Bm)$. Therefore, the additional complexity of the selector during training is

$$O\left(Bnd^2 + Bnd + B^2d + Bm\right), \tag{193}$$

and if the quadratic term of the linear projection is ignored or cached, it can be approximated as

$$O\left(Bnd + B^2d + Bm\right). \tag{194}$$

During inference, if only the selector logit is used to fuse entity scores, without computing NCE, size, and connectivity regularizers, then the additional cost is mainly $O(Bnd^2 + Bnd)$, or $O(Bnd)$ under caching/lightweight projection.

J Training and Inference Complexity

For ease of exposition, suppose that the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has $n = |\mathcal{V}|$ entity nodes and $m = |\mathcal{E}|$ entity-relation edges. If self-loops are added in GCN propagation, we denote $\tilde{m} = m+n$. Let the hidden dimension be d, the number of propagation layers be L, the batch size be B, and the number of pseudo-queries be M. Therefore, one real query together with M pseudo-queries requires $M + 1$ graph reads. Let $\mathbf{M} \in \{0, 1\}^{n\times N_D}$ denote the sparse entity–document association matrix, where N_D is the number of documents, and $\text{nnz}(\mathbf{M})$ is the number of nonzero entity–document links. Let K_e denote the number of top entities used for document projection, and let $\bar{f}$ denote the average number of documents linked to the top entities.

J.1 Offline Structural Feature and Indexing Cost

Let the dimension of node structural features be p_n, the dimension of edge structural features be p_e, and the dimension of graph-level summaries be p_g. In the current implementation, p_n, p_e, and p_g are all small constants.

Given that the adjacency list has been constructed, degrees and average neighbor degrees can be computed in $O(n + m)$ time. Clustering coefficients and the number of common neighbors require computing intersections of neighbor sets, whose complexity can be written as

$$O\left(n + m + \sum_{(u,v)\in\mathcal{E}} \min\{\deg(u), \deg(v)\}\right). \tag{195}$$

In sparse graphs or graphs with bounded average degree, Eq. (195) is approximately $O(n + m)$; in extremely dense graphs, the worst case can reach $O(n^3)$. These structural features and the entity–document matrix can both be precomputed and cached offline, with space cost

$$O\left(np_n + mp_e + p_g + \text{nnz}(\mathbf{M})\right). \tag{196}$$

Since this part does not depend on a specific query, when evaluating multiple queries on the same candidate graph in the self-evolving memory loop, the same set of structural features and entity–document indices can be reused.

J.2 Forward Propagation Complexity of the Structurally Gated GFM

We first consider a single forward propagation for one query on one graph. A standard GCN layer contains two parts: node linear transformation and sparse adjacency aggregation. The cost of node linear transformation is $O(nd^2)$, and the cost of edge-level message aggregation is $O(\tilde{m}d)$. Therefore, the complexity of a standard GCN layer is

$$C_{\text{plain}} = O\left(nd^2 + \tilde{m}d\right). \tag{197}$$

Beyond ordinary message propagation, a structurally gated layer generates a vector gate for each edge. Its message form is

$$\mathbf{m}_{u\to v} = \gg_{uv} \odot W\mathbf{h}_u, \tag{198}$$

where $\mathbf{h}_u$ is the source node representation, $W \in \mathbb{R}^{d\times d}$ is the node linear transformation matrix, $\gg_{uv} \in \mathbb{R}^d$ is the structural gate vector of edge (u, v), and $\odot$ denotes element-wise multiplication. Let d_g denote the encoding dimension of structural features, and let h_g denote the hidden dimension of the gating MLP. If the gate uses four types of inputs, namely source-node structure, target-node structure, edge-pair structure, and graph-level summary, then the gate generation cost can be written as

$$\begin{aligned} C_{\text{gate}} = O\Big(&np_nd_g + mp_ed_g + p_gd_g \\ &+ m(4d_gh_g + h_gd)\Big). \end{aligned} \tag{199}$$

Here, $np_n d_g$ comes from node structural feature encoding, $mp_e d_g$ comes from edge structural feature encoding, $p_g d_g$ comes from graph-level summary encoding, and $m(4d_g h_g + h_g d)$ comes from the per-edge gating MLP. If edge-pair features or graph-level summaries are disabled, the corresponding terms in Eq. (199) can be removed. If d_g and h_g are regarded as being of the same order as d, then gate generation is $O(md^2)$ in the worst case; if the gating MLP is regarded as a small constant-width module, or if low-rank/dimension-wise gating is adopted, it can be approximated as $O(md)$. Therefore, the complexity of a structurally gated layer is

$$C_{\text{gated}} = O\left(nd^2 + \tilde{m}d + C_{\text{gate}}\right). \tag{200}$$

The current implementation supports dual structural prompts: one is a holistic gated branch, and the other is a specific prompt branch. If only a standard GCN is used, the per-layer cost is C_{plain}; if only a structurally gated GCN is used, the per-layer cost is C_{gated}; if one gated branch and one standard branch are used simultaneously, the per-layer cost is approximately $C_{\text{gated}} + C_{\text{plain}}$. Let $\rho_{\text{plain}} \in \{0, 1\}$ denote whether the standard prompt branch is enabled, and let $\rho_{\text{gated}} \in \{0, 1\}$ denote whether the structurally gated branch is enabled. Then the GFM encoding cost for one batch can be uniformly written as

$$C_{\text{enc}}(B) = O\left(BL\left(\rho_{\text{plain}} C_{\text{plain}} + \rho_{\text{gated}} C_{\text{gated}}\right)\right). \tag{201}$$

The factor B appears because, in the current implementation, each query in a batch separately constructs query-conditioned node inputs and performs graph encoding. If query-independent structural gates for a fixed graph are cached during inference, part of the gate cost can be reduced; however, based on the current code implementation, Eq. (201) is a more conservative upper bound.

In the most common simplified analysis, we set $B = 1$, $M = 0$, disable dual branches, and regard the gating MLP as a lightweight constant-width module. Then Eq. (201) degenerates to

$$O\left(L(md + d^2 n)\right), \tag{202}$$

which is exactly the core propagation complexity given in the main text. Here, md corresponds to edge-level messages, structural gating, and sparse aggregation, while $d^2 n$ corresponds to node linear projection.

J.3 Entity Scoring, Selector Regularization, and Document Projection Complexity

After GFM encoding obtains node representations, entity scoring is usually obtained by

$$a_e(q) = \mathbf{h}_e^\top \mathbf{z}_q \tag{203}$$

For one batch, the complexity of this step is

$$C_{\text{score}}(B) = O(Bnd). \tag{204}$$

If the query-conditioned subgraph selector in Appendix I is enabled, then the additional inference-stage cost is

$$C_{\text{sel,infer}}(B) = O(Bnd^2 + Bnd), \tag{205}$$

which can be approximated as $O(Bnd)$ if the node-side projection is cached or a lightweight projection is used. During training, NCE, the size term, and the connectivity term also need to be computed, with additional complexity

$$C_{\text{sel,train}}(B) = O(Bnd^2 + Bnd + B^2 d + Bm). \tag{206}$$

Here, $B^2 d$ comes from the in-batch query–subgraph contrastive matrix, and Bm comes from the edge-level connectivity smoothing term.

Entity-to-document projection is performed by the entity–document matrix $\mathbf{M}$. If full sparse matrix multiplication is used, the complexity is

$$C_{\text{doc,full}}(B) = O\left(B\,\text{nnz}(\mathbf{M})\right). \tag{207}$$

The `IDFWeightedRanker` in the current code belongs to this type: it first constructs IDF weights according to entity occurrence frequency, and then performs sparse matrix multiplication. If top-K_e entity projection is used, conceptually only the inverted lists corresponding to these entities need to be accessed, so the complexity can be written as

$$C_{\text{doc,top}K}(B) = O\left(Bn \log K_e + BK_e \bar{f}\right), \tag{208}$$

where $Bn \log K_e$ comes from top-K_e entity selection, and $BK_e \bar{f}$ comes from accessing the documents linked on average by the top entities. If the final document top-K ranking is performed over all N_D documents, the complexity is $O(BN_D \log K)$; if it is performed only over the candidate document pool, it is $O(BN_{\text{cand}} \log K)$, where $N_{\text{cand}} \ll N_D$.

J.4 Training Complexity

The main costs in the training stage come from GFM forward propagation, the entity-level retrieval loss, the optional selector regularizer, and backpropagation. Let κ_{bw} denote the constant-factor cost of backpropagation relative to forward propagation, which can usually be regarded as a constant between 2 and 3. If the entity-level training loss is BCE, ranking loss, or ListCE, then because the predicted scores of n entities need to be supervised or ranked, the loss computation complexity is

$$C_{\text{loss}}(B) = O(Bn). \tag{209}$$

Therefore, when the selector is not enabled, the complexity of a single training batch is

$$C_{\text{train}} = O\left(\kappa_{\text{bw}}\left[C_{\text{enc}}(B) + C_{\text{score}}(B) + C_{\text{loss}}(B)\right]\right). \tag{210}$$

After enabling the query-conditioned subgraph selector, the training complexity becomes

$$C_{\text{train}}^{\text{sel}} = O\left(\kappa_{\text{bw}}\left[C_{\text{enc}}(B) + C_{\text{score}}(B) + C_{\text{loss}}(B) + C_{\text{sel,train}}(B)\right]\right). \tag{211}$$

If a document-level loss is also explicitly added during training, then entity-to-document projection needs to be additionally performed, with cost $C_{\text{doc,full}}(B)$ or $C_{\text{doc,top}K}(B)$.

J.5 Inference Complexity

The inference stage first performs query encoding, named entity recognition, and entity linking, whose total cost is denoted as $C_{\text{prep}}(q)$. This part depends on the adopted text encoder, NER model, and entity linking model, and does not belong to the graph propagation backbone. Given the prepared query embedding and query entity mask, the core reading complexity of a single query is

$$\begin{aligned} C_{\text{infer}}(q) =& (M+1)\left[C_{\text{enc}}(1) + C_{\text{score}}(1) + C_{\text{sel,infer}}(1) + C_{\text{doc}}(1)\right] \\ &+ C_{\text{fuse}}(M, K), \end{aligned} \tag{212}$$

where M is the number of pseudo-queries, $C_{\text{doc}}(1)$ can take the complexity of full sparse projection or top-K_e inverted projection, and $C_{\text{fuse}}(M, K)$ is the cost of fusing the results from the main query and pseudo-queries. If each query keeps K candidate documents, then the cost of simple weighted merging is

$$C_{\text{fuse}}(M, K) = O((M+1)K \log((M+1)K)), \tag{213}$$

J.6 Space Complexity

The model parameter space mainly comes from the GFM backbone, structural prompts, the structurally gated MLP, selector projections, and text projection layers. If we only discuss graph- related runtime space, offline graph storage requires

$$O\left(n + m + \text{nnz}(\mathbf{M}) + np_n + mp_e + p_g\right). \tag{214}$$

During training, node activations of each layer need to be saved, and the space complexity is on the order of

$$O(BLnd) \tag{215}$$

If structurally gated edge messages are fully materialized, they require $O(md)$ GPU memory; the current implementation adopts edge chunk streaming. Let the chunk size be c, then the peak memory of gated messages can be reduced to

$$O(cd), \tag{216}$$

where $c \ll m$. This is also one of the key engineering designs that makes the current implementation suitable for large-graph reading. If `return_gate` is enabled and the gate vectors of all edges are saved for visualization or interpretation, then the space will rise again to $O(md)$. If document scoring materializes all N_D document scores, it requires $O(BN_D)$ space; if only a candidate document heap is maintained, it can be reduced to $O(BK)$ or $O(BK_e\bar{f})$.

J.7 Complexity Comparison with Related Work

Overall Comparison. From the perspective of complexity, standard dense RAG has the lightest online retrieval cost, but it is difficult to explicitly model cross-document relations; multi-step RAG improves complex reasoning ability through multiple rounds of retrieval, but its cost grows linearly with the number of LLM calls; GraphRAG-style methods shift a large amount of cost to offline graph construction and summary generation; SubgraphRAG reduces online cost through lightweight triple scoring, but its effectiveness depends on the candidate triple set and structural distance features; GFM-RAG and our reader concentrate the main computation on one or a small number of query-conditioned graph propagations. Therefore, when the self-evolving memory loop needs to repeatedly evaluate the retrievability of different written graphs, the advantage of our design lies in the following: each evaluation does not need to start multi-round LLM agentic search, but instead quickly obtains differentiable or scoreable retrieval feedback through a fixed GFM reader, structurally gated propagation, and sparse document projection. This allows the graph writing strategy to perform high-frequency comparison and optimization over a large number of candidate memory graphs.

K Implementation Details of Structured Query Planning

K.1 Detailed definitions of the notation and additional information

In $\mathcal{P}_\omega(q) = \left(\mathcal{E}_{\text{exp}}, \mathcal{A}, C_{\text{rel}}, C_{\text{hard}}, \tau, \{(\tilde{q}_m, \alpha_m, t_m)\}_{m=1}^{M}\right)$, $\mathcal{E}_{\text{exp}}$ acts as a direct anchor for memory (explicit entities); $\mathcal{A}$ maps the brain's multiple representational habits for the same concept (aliases); C_{rel} simulates the relational network in semantic memory; C_{hard} serves as the spatiotemporal and logical boundaries of episodic memory (such as hard constraints like time and location); τ presets the cognitive template of the target memory (answer type); and the pseudo-queries $\tilde{q}_m$ with confidence α_m and intent t_m are analogous to the multiple exploratory recalls conducted in the human mind (Simulated Recall).

K.2 Two-stage Planning: Extraction and Inference

Natural-language questions often compress key retrieval cues into implicit relations, such as "the birthplace of the author", "the publication year of the only mystery novel of a certain work", or "the death date of the father". If the original question is sent as a whole to the entity linker, the system can easily hit only surface entities while missing bridge entities or answer-type constraints. Therefore, the query planner is defined as a structured function

$$\mathcal{P}(q) = \left(\mathcal{E}_{\text{exp}}, \mathcal{A}, C_{\text{rel}}, C_{\text{hard}}, \tau, \{(\tilde{q}_m, \alpha_m)\}_{m=1}^{M}\right). \tag{217}$$

It consists of two stages: *Extractor* 6 extracts explicit entities, aliases, relation clues, hard constraints, and the answer type; *Inferer* 7 generates at most M retrieval intents based on the extraction results.

L Computation Details of Topological Structural Features

L.1 Normalized Structural Graph

Structural features are computed on an undirected, self-loop-free, binarized adjacency matrix $\mathcal{A}_s$:

$$\mathcal{A}_s = \mathbb{I}[(\mathcal{A} + \mathcal{A}^\top) > 0], \qquad \text{diag}(\mathcal{A}_s) = 0. \tag{218}$$

This avoids drastic fluctuations in topological statistics caused by unstable relation-extraction directions. Message propagation can still use the original bidirectional edges or relation-aware graph; structural statistics are only used as gating conditions.

L.2 Node-level Structural Features

For node v, let $\mathcal{N}(v) = \{u : \mathcal{A}_{s,uv} = 1\}$ and $d_v = |\mathcal{N}(v)|$. The node-level features are

$$\phi(v) = \left[\log(1 + d_v), c_v, \kappa_v, \bar{d}_{\mathcal{N}(v)}\right]. \tag{219}$$

The local clustering coefficient is

$$c_v = \begin{cases} \dfrac{2T_v}{d_v(d_v-1)}, & d_v \geq 2, \\ 0, & d_v < 2, \end{cases} \tag{220}$$

Extractor prompt template.

```
You are a retrieval planner for graph-based multi-hop QA.
Question:
{QUESTION}

Extract structured retrieval signals.
Return JSON only with keys:
{
  "explicit_entities": [string],
  "candidate_aliases": {"entity": [alias]},
  "relation_clues": [string],
  "constraints": {},
  "answer_type": "string"
}
Rules: keep entries short, avoid explanations, keep empty fields as [] or {}.
```

Figure 6: Metadata of the case study from HotpotQA.

Inferer prompt template.

```
You are a retrieval planner for graph-based multi-hop QA.
Question:
{QUESTION}

Structured extraction:
{EXTRACTOR_JSON}

Generate at most M retrieval intents that help locate:
- evidence directly supporting the target relation;
- bridge entities required for multi-hop reasoning;
- documents likely to contain the target attribute;
- evidence satisfying temporal, spatial, type, comparison or negation constraints;
- evidence using aliases or alternative mentions.
Return JSON only with keys:
{
  "pseudo_queries": [string],
  "rewriter_confidence": [number]
}
```

Figure 7: Metadata of the case study from HotpotQA.

where T_v is the number of undirected edges inside the neighborhood of v; κ_v is the core number; and the average neighbor degree is

$$\bar{d}_{\mathcal{N}(v)} = \begin{cases} \frac{1}{d_v} \sum_{u \in \mathcal{N}(v)} d_u, & d_v > 0, \\ 0, & d_v = 0. \end{cases} \tag{221}$$

These quantities respectively characterize node frequency, local clustering, core/peripheral position, and neighborhood density. For RAG memory, they correspond to four common structural risks: over-propagation by high-frequency hubs, redundant diffusion inside clustered regions, ignored peripheral bridge entities, and scale mismatch between sparse and dense regions.

L.3 Edge-pair Structural Features

For an undirected structural edge (u, v), the pairwise features are

$$\psi(u, v) = \left[|d_u - d_v|, \mathrm{CN}(u, v), \mathrm{Jac}(u, v) \right], \tag{222}$$

where

$$\mathrm{CN}(u, v) = |\mathcal{N}(u) \cap \mathcal{N}(v)|, \qquad \mathrm{Jac}(u, v) = \frac{|\mathcal{N}(u) \cap \mathcal{N}(v)|}{|\mathcal{N}(u) \cup \mathcal{N}(v)| + \varepsilon}. \tag{223}$$

Degree difference reflects cross-level connections, while common neighbors and Jaccard reflect local community overlap. Based on these features, the gate can distinguish intra-community evidence aggregation edges from cross-community bridge edges.

L.4 Graph-level Summary and Normalization

The graph-level summary concatenates the mean, standard deviation, and density of node features:

$$\mathbf{r}_{\mathcal{G}} = \left[\mathrm{mean}_{v \in \mathcal{V}} \phi(v); \mathrm{std}_{v \in \mathcal{V}} \phi(v); \mathrm{dens}(\mathcal{G}) \right], \tag{224}$$

where

$$\mathrm{dens}(\mathcal{G}) = \begin{cases} \frac{2m_s}{n(n-1)}, & n \geq 2, \\ 0, & n < 2, \end{cases} \tag{225}$$

$n = |\mathcal{V}|$, and m_s is the number of undirected structural edges. To remove graph-size differences, node and edge features are z-scored within each graph, and the graph-level summary computes global mean and standard deviation over the set of training graphs:

$$\bar{\mathbf{r}}_{\mathcal{G}} = \frac{\mathbf{r}_{\mathcal{G}} - \mu_r}{\sigma_r + \varepsilon}. \tag{226}$$

If the standard deviation of a certain dimension is close to zero, we only perform centering to avoid division by an unstable small value.

L.5 Gating Input Encoding

For each message edge $u \to v$, the structural gate reads the source node, target node, pairwise features, and graph-level summary:

$$\bar{\phi}(u) = \mathrm{NormNode}(\phi(u)), \quad \bar{\psi}(u,v) = \mathrm{NormPair}(\psi(u,v)), \tag{227}$$

$$u_u^{(l)} = E_n^{(l)}(\bar{\phi}(u)), \qquad u_v^{(l)} = E_n^{(l)}(\bar{\phi}(v)), \tag{228}$$

$$v_{uv}^{(l)} = E_p^{(l)}(\bar{\psi}(u,v)), \qquad r_{\mathcal{G}}^{(l)} = E_g^{(l)}(\bar{\mathbf{r}}_{\mathcal{G}}). \tag{229}$$

The encoders E_n, E_p, E_g are all two-layer MLPs. The concatenated gating input is

$$\mathbf{z}_{uv}^{(l)} = [\mathbf{u}_u^{(l)}; \mathbf{u}_v^{(l)}; \mathbf{v}_{uv}^{(l)}; \mathbf{r}_{\mathcal{G}}^{(l)}]. \tag{230}$$

The gate itself is a vector rather than a scalar:

$$\mathbf{g}_{uv}^{(l)} = \mathbf{1} + \delta \tanh\left(\mathrm{MLP}_g^{(l)}(\mathbf{z}_{uv}^{(l)})\right), \qquad \delta = 0.1. \tag{231}$$

The last layer of the gating MLP is initialized to zero, so initially $\mathbf{g}_{uv}^{(l)} = \mathbf{1}$. At the beginning of training, the model does not destroy the original propagation scale; the learned structural bias gradually emerges in a residual manner.

L.6 Message Propagation with Normalized Weights

Let $\tilde{\mathbb{E}}$ be the edge set after adding self-loops. Structural gates are used for non-self-loop edges, and unit gates are used for self-loops. The GCN normalization coefficient is

$$\eta_{uv} = \frac{w_{uv}}{\sqrt{\tilde{d}_u \tilde{d}_v}}, \qquad \tilde{d}_v = \sum_{u:(u,v)\in\tilde{\mathbb{E}}} w_{uv}, \tag{232}$$

where w_{uv} defaults to 1, but can also come from edge weights. The propagation at layer l is

$$\mathbf{m}_{u\to v}^{(l)} = \eta_{uv}\, \mathbf{g}_{uv}^{(l)} \odot W^{(l)} \mathbf{h}_u^{(l-1)}, \tag{233}$$

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{b}^{(l)} + \sum_{u:(u,v)\in\tilde{\mathbb{E}}} \mathbf{m}_{u\to v}^{(l)}\right). \tag{234}$$

The multi-layer wrapper also contains inter-layer residuals: when $l > 1$ and the dimensions are consistent,

$$\mathbf{H}^{(l)} \leftarrow \mathbf{H}^{(l)} + \mathbf{H}^{(l-1)}. \tag{235}$$

This residual and Eq. (231) form a dual stability mechanism: the former stabilizes deep propagation, while the latter stabilizes structural modulation.

L.7 Chunked Gating and GPU Memory Complexity

Explicitly storing all gates requires $O(|\mathbb{E}|d)$ GPU memory. For large graphs, gates are computed by edge chunks:

$$\mathbb{E} = \bigcup_{b=1}^{B_e} \mathbb{E}_b, \qquad |\mathbb{E}_b| \le C_e. \tag{236}$$

Each edge chunk sequentially executes

$$\mathbf{g}_b \to \mathbf{m}_b \to \mathrm{scatter\_add}(\mathbf{m}_b), \tag{237}$$

and immediately releases the intermediate gate tensor. Online GPU memory is reduced from $O(|\mathbb{E}|d)$ to $O(C_e d)$, while the time complexity remains linear, $O(|\mathbb{E}|d)$. This is especially important for self-evolving memory, because the same reader needs to repeatedly evaluate candidate graphs produced by different writers.

M Pretraining Objective and Augmented Views

M.1 GraphCL View Construction

The goal of the pretraining stage is to learn cross-graph transferable structural–semantic propagation, rather than fitting specific question-answering labels. Given the original graph view $(\mathcal{G}_0, X_0)$, we construct two augmented views $(\mathcal{G}_1, X_1)$, $(\mathcal{G}_2, X_2)$ and one negative feature view $(\mathcal{G}_0, X^-)$. The augmentation types include edge perturbation, feature masking, node perturbation, and subgraph sampling; let the augmentation operators be $\mathcal{A}_1, \mathcal{A}_2$, then

$$(\mathcal{G}_j, X_j) = \mathcal{A}_j(\mathcal{G}_0, X_0), \qquad j \in \{1,2\}. \tag{238}$$

If structural gating is enabled, each view precomputes its own node structural features, edge-pair features, and graph-level summary; the negative feature view shares the base graph structure, but its node features are shuffled or replaced.

M.2 Graph-level Contrastive Objective

The encoder outputs four sets of node representations:

$$H_0 = f_\theta(X_0, \mathcal{G}_0), \quad H_1 = f_\theta(X_1, \mathcal{G}_1), \quad H_2 = f_\theta(X_2, \mathcal{G}_2), \quad H^- = f_\theta(X^-, \mathcal{G}_0). \tag{239}$$

The graph readout of each augmented view is

$$c_j = \mathrm{sigmoid}\left(\frac{1}{|\mathcal{V}_j|}\sum_{v\in\mathcal{V}_j} H_{j,v}\right), \qquad j \in \{1,2\}. \tag{240}$$

The bilinear discriminator

$$D(c,h) = h^\top W_D c \tag{241}$$

determines whether the node representation comes from the same graph semantics. The pretraining loss is

$$\mathcal{L}_{\mathrm{GCL}} = \frac{1}{2}\sum_{j=1}^{2}\left[\mathrm{BCE}\left(D(c_j, H_0), \mathbf{1}\right) + \mathrm{BCE}\left(D(c_j, H^-), \mathbf{0}\right)\right]. \tag{242}$$

When edge-level gating is enabled, traditional static structural prompts are neutralized into identity mappings to avoid scale confusion caused by two sets of structural modulations acting simultaneously; the structural bias is mainly carried by the target edge's $\mathbf{g}_{uv}^{(l)}$.

M.3 Feature Alignment Layer

When the input dimensions produced by different graphs or different text encoders are consistent but their distributions have large shifts, the feature alignment layer can be enabled:

$$\mathrm{Align}(x) = \mathrm{Dropout}\left(\mathrm{LayerNorm}\left(\mathrm{PReLU}(W_a x + b_a)\right)\right). \tag{243}$$

W_a is initialized as the identity matrix, and b_a is initialized as zero. Therefore, this layer is initially an approximately identity transformation; after training, it absorbs inter-graph feature-scale differences without changing the core structure of the graph propagator.

N Supervised Fine-tuning Objective

N.1 Entity-level Supervision

For each question q_b, the data provide a supporting-entity mask $y_{b,e} \in \{0,1\}$. The model outputs entity logits $a_{b,e}$. The weighted BCE is defined as

$$\mathcal{L}_{\text{bce}} = \frac{1}{B}\sum_{b=1}^{B} \frac{\sum_e w_{b,e}\ \text{BCEWithLogits}(a_{b,e}, y_{b,e})}{\sum_e w_{b,e} + \varepsilon}. \tag{244}$$

Positive weights are uniformly normalized within the positive set; if the adversarial temperature T_a is enabled for negative weights, they are computed by applying softmax to the current model scores:

$$w^-_{b,e} = \frac{\exp(a_{b,e}/T_a)}{\sum_{v:y_{b,v}=0}\exp(a_{b,v}/T_a)}, \qquad y_{b,e} = 0. \tag{245}$$

If $T_a = 0$, the negative weights degenerate into a uniform distribution. This design makes training focus more on high-scoring hard negatives, rather than being dominated by a large number of obviously irrelevant entities.

N.2 Multi-positive List Cross-Entropy

Using only BCE treats each entity as an independent binary classification problem, lacking the constraint that "supporting entities should collectively rank near the top of the same candidate list". To this end, we introduce a multi-positive list loss. Let

$$p_{b,e} = \frac{\text{sigmoid}(a_{b,e})}{\sum_v \text{sigmoid}(a_{b,v}) + \varepsilon}. \tag{246}$$

If sample b has at least one supporting entity, the list loss is

$$\mathcal{L}_{\text{list}} = -\frac{1}{|\mathcal{B}_+|}\sum_{b\in\mathcal{B}_+}\frac{1}{|Y_E(q_b)|}\sum_{e\in Y_E(q_b)}\log(p_{b,e}+\varepsilon). \tag{247}$$

Samples with empty supporting-entity sets are skipped. The final entity fine-tuning objective is

$$\mathcal{L}_{\text{ent}} = \lambda_{\text{bce}}\mathcal{L}_{\text{bce}} + \lambda_{\text{list}}\mathcal{L}_{\text{list}}, \qquad (\lambda_{\text{bce}}, \lambda_{\text{list}}) = (0.3, 0.7). \tag{248}$$

N.3 Optional Document-level Supervision

If the training configuration provides a document-level loss, entity logits are first projected into document logits:

$$\tilde{S}_b = a_b^\top \mathbf{M}, \tag{249}$$

and then the same type of BCE or list loss is computed with the supporting-document mask $z_{b,i}$. This term is suitable for tasks where entity annotations are noisy but the document support set is reliable; if it is not enabled, training is entirely driven by the entity-level support set, and document ranking is obtained through projection only during inference or validation.

O Memory Writer Implementation Details

O.1 The Markov Decision Process for Multi-turn Graph Construction

Specifically, the training of our graph constructor is implemented through VeRL's multi-turn GRPO loop. The state machine of the interactor can be abstracted as a finite-horizon MDP:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \rho_0, H). \tag{250}$$

Given a sample x, at round t, the state can be written as

$$s_t = (q, \mathcal{G}_t, \mathcal{D}_t^{\text{proc}}, \mathcal{D}_t^{\text{rem}}, \zeta_t), \tag{251}$$

where $\mathcal{G}_t$ is the current partially written graph, $\mathcal{D}_t^{\text{proc}}$ and $\mathcal{D}_t^{\text{rem}}$ denote the processed and remaining documents, respectively, and ζ_t is an interaction control flag, such as whether the process is still in the graph-construction stage or has already switched to the RAG stage. The action is generated by the language model in JSON format:

$$a_t \sim \pi_\theta(\cdot \mid s_t), \tag{252}$$

and is restricted to two types of legal actions:

(1) **Triple action**: output a JSON array, where each element is of the form {`subject`, `relation`, `object`}, representing the set of facts $\mathcal{T}_t$ written in the current round;
(2) **Termination action**: after graph construction is completed, output a JSON object carrying the terminal fields required by the reader side, such as `answer`, `recall`, `precision`, `deducible`, and so on.

In implementation, the environment first checks whether the action can be parsed by `json_repair`, and strictly cleans the triples: items with missing keys, empty strings, or non-dictionary entries are all removed. If illegal JSON is output during the graph-construction stage, the interaction terminates immediately and returns zero reward; if legal triples are output, the environment proceeds to the next round and returns a round-level format reward. The corresponding environment transition can be written as

$$s_{t+1} = P(s_t, a_t) = \begin{cases} (q, \mathcal{G}_t \oplus \mathcal{T}_t, \mathcal{D}_t^{\text{proc}} \cup \{d_t\}, \\ \quad \mathcal{D}_t^{\text{rem}} \setminus \{d_t\}, \zeta_{t+1}) & a_t \text{ is legal,} \\ (q, \mathcal{G}_t, \mathcal{D}_t^{\text{proc}}, \mathcal{D}_t^{\text{rem}}, \text{STOP}) & a_t \text{ is illegal,} \\ (q, \mathcal{G}_t, \mathcal{D}_t^{\text{proc}}, \mathcal{D}_t^{\text{rem}}, \text{RAG}) & a_t \text{ triggers reading.} \end{cases} \tag{253}$$

Iterative and non-iterative writing. Two strategies are supported. In the non-iterative mode, the model reads the entire context $\mathcal{D}$ at once and outputs all triples. In the iterative mode, the environment reads the documents segment by segment in document order, and in each round the model is only allowed to write triples for the current document. After all documents have been processed, the environment then switches to the RAG stage. If $\mathcal{T}_i$ denotes the set of triples output for document d_i, then the final graph constructed in the iterative mode is $\mathcal{G} = \bigoplus_{i=1}^{m} \mathcal{T}_i$, where $\oplus$ denotes edge-set union and node deduplication. We adopt the iterative strategy by default, because it decomposes the long-context problem into a sequence of local writing decisions, significantly reducing the difficulty of performing global planning in advance. At the same time, it also allows the source document of each triple to be precisely recorded, providing explicit source edges for subsequent text-graph retrieval.

Constructing text-graph memory from output triples. To enable the frozen retriever to operate under the **graph-guided text retrieval** setting, the environment does not directly pass the raw triple strings to the retriever. Instead, it first constructs a text graph with document nodes:

$$\mathcal{G} = (\mathcal{V}_e \cup \mathcal{V}_d, \mathcal{E}_{ee} \cup \mathcal{E}_{ed}), \tag{254}$$

where the entity node set $\mathcal{V}_e$ comes from the subjects and objects in the triples, and the document node set $\mathcal{V}_d = \{d_1, \ldots, d_m\}$ corresponds to the original documents in the context. The entity-entity edges are defined as

$$\mathcal{E}_{ee} = \{(u, r, v) \mid (u, r, v) \in \mathcal{T}\}, \tag{255}$$

and the entity-document source edges are defined as

$$\mathcal{E}_{ed} = \{(u, \texttt{source}, d_i), (v, \texttt{source}, d_i) \mid (u, r, v) \in \mathcal{T}_i\}. \tag{256}$$

In the iterative mode, the source edges are explicit, because the environment already knows that the triples in each round come from the current document. In the non-iterative mode, we use a heuristic alignment method based on tokenizer token overlap to map each triple to the most similar document. The significance of this design is that, after separating writing from reading, the graph constructor is only responsible for deciding "what to write into memory"; as for how the reader aggregates entities on the graph and retrieves documents, this is entirely determined by the frozen f_ϕ.

Frozen GFM retrieval environment. When training the graph constructor, the reader f_ϕ is fixed as the already trained GFM retriever. Let the entity set be $\mathcal{V}_e = \{e_1, \ldots, e_n\}$ and the document set be $\mathcal{V}_d = \{d_1, \ldots, d_M\}$. We then construct:

(1) the relation-edge index $\mathbf{E}$ with both forward and reverse directions, together with the relation types $\mathbf{r}$;
(2) the sparse entity-document matrix $\mathbf{M} \in \{0, 1\}^{n \times M}$, where $M_{ij} = 1$ if entity e_i appears in document d_j;
(3) the question-related entity mask $\mathbf{m}_q \in \{0, 1\}^n$, which is obtained preferentially through lexical matching with the question; if lexical matching fails, it falls back to a heuristic seed set ranked by entity degree.

After encoding the question as a vector $\mathbf{q}$ and the relation names as a matrix $\mathbf{R}$, the frozen GFM forward pass computes the entity relevance scores:

$$\mathbf{s}_e = f_\phi(\mathcal{G}, \mathbf{q}, \mathbf{m}_q; \phi) \in \mathbb{R}^n. \tag{257}$$

The entity scores are then projected into document scores. Let $\mathbf{M}_{\text{Top-}k}(\mathbf{s}_e)$ denote the masking operation that retains only the top-K entity scores, and let $\mathbf{w}_{\text{idf}}$ denote the inverse-frequency weights defined according to the document frequency of each entity. The four document-scoring modes can be written uniformly as

$$\tilde{\mathbf{s}}_e = \begin{cases} \mathbf{s}_e, & \texttt{raw}, \\ \mathbf{M}_{\text{Top-}k}(\mathbf{s}_e), & \texttt{topk}, \\ \mathbf{w}_{\text{idf}} \odot \mathbf{s}_e, & \texttt{idf}, \\ \mathbf{w}_{\text{idf}} \odot \mathbf{M}_{\text{Top-}k}(\mathbf{s}_e), & \texttt{idf\_topk}, \end{cases} \tag{258}$$

and the document scores are obtained by

$$\mathbf{s}_d = \mathbf{M}^\top \tilde{\mathbf{s}}_e. \tag{259}$$

We then take Top-$k(\mathbf{s}_d)$ as the retrieval result. In actual use, we also enable `init_entities_weight`, that is, during the GFM forward pass, a $1/f(e)$ weight is applied to high-frequency entities to suppress the dominance of entities connected to too many documents in the retrieval results.

P Additional Detailed Experimental Results

The results of retrieval performance on multi-hop QA benchmarks are in Table 9.

The results on AmazonQA are in Table 10.

The HaluMem results are shown in Table 11.

P.1 Path Interpretations

We provide path interpretations of SAGE for multi-hop reasoning in Table 12. The importance of each path to the final prediction can be measured by the partial derivative of the prediction score with respect to the triples at each reasoning layer. The top-k path interpretations are then obtained by selecting the top-k longest paths with beam search.

As shown in Table 12, SAGE successfully identifies the answer by connecting two key constraints in the question: the person who presented the Australia 2022 FIFA World Cup bid and the person born on October 22, 1930. Specifically, the first path starts from the entity "the bid for the 2022 FIFA World Cup" and follows the inverse relation of "was one of the representatives of" to reach "Frank Lowy". Then, through an entity-equivalence relation, it links "Frank Lowy" to "Sir Frank P. Lowy", whose birth date is "22 October 1930". The second path verifies the reasoning in the reverse direction by starting from the birth date and tracing back to the representative of the World Cup bid. These paths demonstrate that SAGE can effectively align different surface forms of the same entity and integrate multiple question constraints within a single-step retrieval process, showing its ability to perform interpretable multi-hop reasoning.

Q Dataset Details

Table 13 summarizes the details of each dataset.

General and Multi-hop QA.. We first evaluate SAGE on a set of general open-domain and multi-hop QA benchmarks that stress different aspects of retrieval-augmented reasoning. `NQ-Open` is derived from Natural Questions and is widely used as a standard open-domain short-answer QA benchmark; it evaluates whether a system can retrieve and ground factual answers from a large Wikipedia-scale corpus. `PopQA` complements NQ by focusing on entity-centric factual questions whose subjects span different popularity levels, making it particularly useful for testing whether a retrieval or memory system can recover long-tail factual knowledge rather than relying only on parametric memorization. `HotpotQA` contains Wikipedia-based multi-hop questions with sentence-level supporting facts, allowing us to evaluate not only answer correctness but also whether the system can recover bridge evidence and produce interpretable reasoning chains. `2WikiMultiHopQA` further stresses structured multi-hop reasoning by combining Wikipedia text with Wikidata-derived relations and providing evidence paths for 2–4 hop questions. Finally, `MuSiQue` is designed to reduce shortcut reasoning by composing connected single-hop questions into 2–4 hop questions, making it a strong testbed for evaluating whether SAGE can retrieve and integrate multiple pieces of evidence in a genuinely compositional manner.

Table 9: Results of retrieval performance on multi-hop QA benchmarks. We report document-level Recall (%) at top-2 and top-5. Best results are in bold and runner-ups are underlined. The darker the cell, the better.

Dataset	HotpotQA		MuSiQue		2WikiMultiHopQA		Avg. Rank
Method	R@2	R@5	R@2	R@5	R@2	R@5	
BM25 (▷ *SIGIR'94*)	55.4	72.2	32.3	41.2	51.8	61.9	18.2
Contriever (▷ *TMLR'22*)	57.2	75.5	34.8	46.6	46.6	57.5	17.2
GTR (▷ *EMNLP'22*)	59.4	73.3	37.4	49.1	60.2	67.9	13.8
ColBERTv2 (▷ *NAACL'22*)	64.7	79.3	37.9	49.2	59.2	68.2	11.5
RAPTOR (▷ *ICLR'24*)	58.1	71.2	35.7	45.3	46.3	53.8	17.7
Proposition (▷ *EMNLP'24*)	58.7	71.1	37.6	49.3	56.4	63.1	14.8
GraphRAG (▷ *arXiv'24*)	58.3	76.6	35.4	49.3	61.6	77.3	12.2
G-Retriever (▷ *NeurIPS'24*)	53.3	65.5	38.8	45.1	60.8	67.8	15.7
LightRAG (▷ *arXiv'24*)	38.8	54.7	24.8	34.7	45.1	59.1	20.5
HippoRAG (▷ *NeurIPS'24*)	60.1	78.5	41.2	53.2	68.4	87.0	8.7
HippoRAG 2 (▷ *ICML'25*)	80.5	88.1	47.0	56.7	**88.9**	90.1	2.4
SubgraphRAG (▷ *ICLR'25*)	61.5	73.0	42.1	49.3	70.7	85.5	9.7
PropRAG (▷ *EMNLP'25*)	**81.9**	88.0	**47.7**	**57.9**	87.9	90.1	**1.9**
GFM-RAG (▷ *NeurIPS'25*)	75.6	**89.6**	43.5	57.6	79.1	92.4	2.9
FLARE (▷ *EMNLP'23*)	73.1	81.3	44.3	55.1	67.1	73.1	6.5
Adaptive-RAG (▷ *NAACL'24*)	61.0	76.4	35.1	44.7	44.7	61.4	16.6
BM25 + IRCoT (▷ *ACL'23*)	65.6	79.0	34.2	44.7	61.2	75.6	12.4
Contriever + IRCoT (▷ *ACL'23*)	65.9	81.6	39.1	52.2	51.6	63.8	10.7
ColBERTv2 + IRCoT (▷ *ACL'23*)	67.9	82.0	41.7	53.7	64.1	74.4	7.2
HippoRAG + IRCoT (▷ *ACL'23*)	67.0	83.0	45.3	57.6	75.8	**93.9**	3.6
SAGE (ours)	65.1	77.6	43.2	53.1	83.6	88.6	7.0

E-commerce Review-based QA.. We use AmazonQA to evaluate SAGE in a practical, noisy, user-generated e-commerce setting. Unlike Wikipedia-style QA benchmarks, AmazonQA consists of real product questions, community answers, product reviews, and product metadata, and includes answerability annotations indicating whether a question can be answered from available reviews. This makes it a suitable benchmark for testing whether a memory system can identify useful evidence from noisy review collections, distinguish answerable from unanswerable questions, and synthesize grounded answers from multiple user-generated snippets. From the perspective of self-evolving memory, AmazonQA is especially valuable because the system must learn which review facts, product attributes, and user opinions are worth indexing for future retrieval, rather than simply matching a question to a clean encyclopedic passage.

Long-term Agent Memory. To move beyond conventional RAG evaluation, we further evaluate SAGE on long-term agent memory benchmarks. LongMemEval is designed to assess the long-term memory abilities of chat assistants over extended multi-session interaction histories. It covers five core memory abilities: information extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention. This benchmark directly tests whether SAGE can retrieve sparse but relevant memory traces from long histories, combine evidence across sessions, respect temporal order, and update previously stored information when new interactions supersede old memories. We use HaluMem as a complementary benchmark for evaluating hallucination in memory systems. Rather than only measuring end-to-end QA accuracy, HaluMem decomposes memory evaluation into memory extraction, memory updating, and memory question answering, thereby revealing at which operational stage hallucinations, omissions, or conflicts arise. This is particularly important for our setting because errors introduced during graph construction or memory updating may propagate to graph-guided retrieval and final answer generation.

Evaluation Rationale. Together, these datasets form a progressively broader evaluation suite. NQ and PopQA test factual open-domain retrieval; HotpotQA, 2WikiMultiHopQA, and MuSiQue test

Table 10: Performance of representative baselines on the original AmazonQA full-test protocol. BLEU-1/2/3/4 are denoted as B-1/2/3/4, and R denotes ROUGE. Best results are in bold and runner-ups are underlined. Only rows marked with 0-shot are our zero-shot transfer results; baseline rows and trained variants are not marked as zero-shot.

Zero-shot setting applies only to Ours rows marked with 0-shot on AmazonQA.

Method	B-1	B-2	B-3	B-4	R
Heuristic baselines from the original AmazonQA protocol					
Random Sentence (▷ *IJCAI'19*)	78.56	63.95	44.37	29.87	49.12
Top-1 using IR (▷ *IJCAI'19*)	89.49	74.80	56.76	43.52	61.48
Top-1 Using BLEU (▷ *IJCAI'19*)	**92.74**	**78.43**	**60.91**	**48.08**	**62.68**
Top-1 Helpfulness (▷ *IJCAI'19*)	20.66	19.78	16.39	12.54	37.01
Top-1 Wilson Score (▷ *IJCAI'19*)	20.74	19.84	16.44	12.58	37.26
Neural baseline from the original AmazonQA protocol					
R-Net (▷ *IJCAI'19*)	47.04	40.32	31.48	23.92	40.22
Human answers under the original AmazonQA protocol					
Amazon User Community (▷ *IJCAI'19*)	80.88	68.86	54.36	42.01	62.18
Expert (Spans) (▷ *IJCAI'19*)	68.33	57.79	44.61	34.43	51.09
Expert (Descriptive) (▷ *IJCAI'19*)	53.67	46.56	37.81	30.76	53.31
Our method					
Ours (0-shot) 0-shot	61.84 0-shot	49.36 0-shot	37.82 0-shot	28.41 0-shot	46.73 0-shot
Ours (trained)	74.92	61.58	47.63	35.86	54.92
Ours +1 round	82.76	68.91	52.74	39.68	58.83

Table 11: Results on HaluMem-Medium. We report memory extraction metrics, memory updating metrics, and memory question-answering metrics. R denotes Recall, W-R denotes Weighted Recall, T-P denotes Target Memory Precision, Acc. denotes Memory Accuracy, FMR denotes False Memory Resistance, F1 denotes Memory Extraction F1-score, C denotes Correct Rate, H denotes Hallucination Rate, and O denotes Omission Rate. For R, W-R, T-P, Acc., FMR, F1, and C, higher is better; for H and O, lower is better. Best results are in bold and runner-ups are underlined. The darker the cell, the better. For systems whose public reports only provide a subset of metrics, missing entries are denoted by "–". Only rows marked with 0-shot are our zero-shot results; baseline rows and trained SAGE rows are not marked as zero-shot.

Zero-shot setting applies only to SAGE variants marked with 0-shot on HaluMem-Medium.

Dataset	Memory Extraction						Memory Updating			Memory QA		
Method	R↑	W-R↑	T-P↑	Acc.↑	FMR↑	F1↑	C↑	H↓	O↓	C↑	H↓	O↓
Memory-system baselines from the original HaluMem benchmark												
Memobase (https://github.com/memodb-io/memobase)	14.55	25.88	**92.24**	32.29	**80.78**	25.13	5.20	0.55	94.25	35.33	29.97	34.71
Supermemory (https://github.com/supermemoryai/supermemory)	41.53	64.76	90.32	60.83	51.77	56.90	16.37	1.15	82.47	54.07	22.24	23.69
Mem0 (▷ *arXiv'25*)	42.91	65.03	86.26	**60.86**	56.80	57.31	25.50	0.45	74.02	53.02	19.17	27.81
Zep (▷ *arXiv'25*)	–	–	–	–	–	–	47.28	**0.42**	52.31	55.47	21.92	22.62
MemOS (▷ *arXiv'25*)	**74.07**	**84.81**	86.25	59.55	44.94	**79.70**	**62.11**	**0.42**	**37.48**	**67.23**	**15.17**	**17.59**
SAGE (ours, 0-shot) 0-shot	13.12	20.91	31.36	22.80	32.59	19.38	21.67	1.61	84.53	30.14	33.68	36.17
SAGE (ours, trained)	16.42	29.36	71.88	35.41	44.96	28.52	7.34	0.68	91.98	38.26	28.73	33.01
SAGE +1 round	20.18	35.74	71.02	40.63	40.28	33.47	10.86	0.76	88.38	42.91	26.64	30.45

multi-hop evidence composition; AmazonQA evaluates noisy real-world review memory in an e-commerce domain; LongMemEval tests long-horizon interactive memory; and HaluMem diagnoses operation-level hallucinations in memory systems. This combination allows us to evaluate SAGE not merely as a retrieval-augmented QA pipeline, but as a self-evolving memory system that must decide what to store, how to organize stored information, how to retrieve

Table 12: Path interpretations of SAMGPT for multi-hop reasoning, where r^{-1} denotes the inverse of original relation.

Question	Which man who presented the *Australia 2022 FIFA World Cup bid* was born on *October 22, 1930*?
Answer	Frank Lowy
Sup. Doc.	["Frank Lowy", "Australia 2022 FIFA World Cup bid"]
Paths	1: (the bid for the 2022 fifa world cup, was one of the representatives of^{-1}, frank lowy) → (frank lowy, equivalent, sir frank p lowy) → (sir frank p lowy, was born on, 22 october 1930) 2: (22 october 1930, was born on^{-1}, sir frank p lowy) → (sir frank p lowy, equivalent, frank lowy) → (frank lowy, was one of the representatives of, the bid for the 2022 fifa world cup)

Table 13: Dataset statistics and evaluation scenarios. We evaluate SAGE on three complementary categories: general and multi-hop QA, practical e-commerce review QA, and long-term agent memory. "Train/Dev/Test" denotes the standard split when available. For benchmark-only datasets without a conventional supervised training split, we report the total number of evaluation instances or benchmark scale.

Category	Dataset	Scale / Split	Evidence Source	Task Type	Key Capabilities	Main Metrics
General / Multi-hop QA	NQ-Open (▷; ▷)	79,168 / 8,757 / 3,610	English Wikipedia	Open-domain short-answer QA	Factual retrieval; entity-level knowledge access; open-domain answer generation	EM / F1 / Acc.; Recall@k
	PopQA (▷)	14,267 QA pairs	Wikidata triples + Wikipedia page-view popularity	Entity-centric open-domain QA	Long-tail factual recall; parametric vs. non-parametric memory; retrieval under entity popularity shift	Acc. / EM; long-tail breakdown
	HotpotQA (▷)	90,447 / 7,405 / 7,405	Wikipedia paragraphs; 10-paragraph distractor setting	Explainable 2-hop QA	Bridge-entity recovery; comparison reasoning; sentence-level supporting facts	Answer EM/F1; Support EM/F1; Joint EM/F1
	2WikiMultiHopQA (▷)	167,454 / 12,576 / 12,576	Wikipedia + Wikidata; 10 passages per instance	2–4 hop multi-hop QA	Reasoning-path recovery; comparison, bridge, and bridge-comparison reasoning	Answer EM/F1; Evidence / path recall
	MuSiQue (▷)	19,938 / 2,417 / 2,459 (24,814 total)	Composed single-hop QA over textual passages	2–4 hop connected multi-hop QA	Connected reasoning; shortcut-resistant evidence aggregation; multi-hop compositionality	Answer EM/F1; Support / evidence recall
E-commerce Review QA	AmazonQA (▷)	923K questions; 3.6M answers; 14M reviews; 156K products	Amazon product reviews, questions, answers, and product metadata	Review-based QA with answerability annotation	Noisy review retrieval; answerable / unanswerable detection; evidence synthesis from user-generated reviews	BLEU / ROUGE; answerability Acc./F1; groundedness
Long-term Agent Memory	LongMemEval (▷)	500 eval. instances per file; S: ∼115K tokens / 30–40 sessions; M: ∼1.5M tokens / ∼500 sessions; Oracle: evidence sessions only	Long multi-session human–AI chat histories	Long-term interactive memory QA	Information extraction; multi-session reasoning; temporal reasoning; knowledge update; abstention	Overall Acc.; category-wise Acc.; context tokens; latency
	HaluMem (▷)	Medium: 20 users, 30,073 dialogue rounds, ∼160K tokens/user, 14,948 memory points, 3,467 QA pairs; Long: 53,516 rounds, ∼1M tokens/user	Synthetic long-term human–AI interaction histories with memory points and multi-type questions	Operation-level memory hallucination benchmark	Memory extraction; memory updating; memory QA; hallucination, omission, and conflict propagation across memory operations	Extraction R/P/F1; Updating C/H/O; QA C/H/O

it under different query conditions, and how to update or suppress unreliable memories over time.

R Baselines and Metrics

Baselines. We evaluate SAGE against state-of-the-art baselines, including their combined variants, which are grouped into **four** categories:

- ***Base LLM:*** GPT-4o-mini [17].
- ***Single-step RAGs:*** including BM25 [35], Contriever [18], GTR [30], ColBERTv2 [36], RAPTOR [37], and Proposition [4].
- ***Graph-enhanced RAGs:*** including GraphRAG [8], G-Retriever [13], LightRAG [9], HippoRAG [11], HippoRAG 2 [11], SubgraphRAG [24], PropRAG [41], and the closely related GFM-RAG [28].
- ***Multi-step RAGs:*** IRCoT [39], FLARE [20], and Adaptive-RAG [19].

In particular, IRCoT [39] is a general multi-step reasoning framework that can be integrated with non-iterative retrievers, allowing both single-step RAG and graph-based methods to conduct multi-hop reasoning through interleaved retrieval and generation. Table **??** presents a comprehensive comparison between all baselines and SAGE.

Metrics. To evaluate retrieval quality, we report Recall@2 and Recall@5 for both retrieved entities and documents, denoted as $R@2/5_E$ and $R@2/5_D$, respectively.. For end-to-end QA evaluation, we use standard metrics, including Exact Match (EM), F1 score, Precision (P), and Recall (R), in the main experiments to comprehensively measure answer correctness and coverage.

Limitations

SAGE treats graph memory as a dynamic substrate for writing, reading, and self-evolution, but its effectiveness still depends on the quality of entity extraction, relation writing, source anchoring, and reader feedback. Errors introduced during graph construction may propagate to retrieval and final answer generation, especially in long-term memory settings involving temporal updates, conflicting user preferences, or sparse evidence. Our experiments show promising results across multi-hop QA, open-domain retrieval, review-based QA, and long-term agent-memory benchmarks, but the current system still leaves room for improvement on memory updating, high-coverage extraction, and hallucination control in more realistic deployments. The theoretical analysis also relies on assumptions such as bounded graph drift, aggregate signal propagation, and local Lipschitz stability, which provide useful intuition but may not capture all failure modes of large-scale, noisy, continuously evolving memory graphs.

Broader Impact

This work may have positive societal impact by improving the reliability and grounding of long-horizon language agents. A structure-aware and self-evolving memory system can help agents recover evidence chains from fragmented cues, maintain more consistent long-term interactions, and reduce unsupported answers in applications such as knowledge assistance, research support, customer support, and review-based question answering. At the same time, long-term agent memory raises important risks. If deployed on personal or sensitive interaction histories, such systems may store private information, infer user preferences, preserve outdated or incorrect memories, or enable profiling and surveillance. Incorrect graph writes or retrieval failures may also lead to confidently grounded but wrong answers. Practical deployments should therefore use consent-based data collection, data minimization, access control, deletion and forgetting mechanisms, provenance tracking, auditing, and human oversight for high-stakes use cases.

Compute Resources

All experiments were run on a server equipped with 8 NVIDIA A100 GPUs. The main computational cost of SAGE comes from graph-memory construction, GFM-based graph propagation, selector regularization, and entity-to-document projection. Appendix J analyzes the training and inference complexity in terms of the number of graph nodes n, edges m, hidden dimension d, propagation layers L, batch size B, pseudo-queries M, and entity-document links. In our implementation, structural features and entity-document indices can be precomputed and cached, while edge-level gates are computed in chunks to reduce peak GPU memory from $O(|E|d)$ to $O(C_e d)$ for chunk size C_e. The dominant inference cost is one or a small number of query-conditioned graph propagations followed by sparse document projection, making the reader suitable for repeated evaluation inside the self-evolving writer–reader loop.

Licenses and Existing Assets

This paper uses existing public benchmarks and baselines, including NQ-Open, PopQA, HotpotQA, 2WikiMultiHopQA, MuSiQue, AmazonQA, LongMemEval, HaluMem, BM25, Contriever, GTR, ColBERTv2, RAPTOR, GraphRAG, G-Retriever, LightRAG, HippoRAG, HippoRAG 2, SubgraphRAG, PropRAG, GFM-RAG, IRCoT, FLARE, and Adaptive-RAG. We cite the original papers or repositories for these assets and use them only for research evaluation under their stated licenses and terms of use. We do not redistribute modified versions of the datasets beyond the preprocessing scripts and instructions needed for reproducibility. The released code is intended for research use and includes documentation for environment setup, data preparation, training, and evaluation.

GenAI Usage Disclosure

AI tools were used only to assist with LaTeX formatting conversion, anonymized-review formatting, and compilation-error fixing for this workshop submission. They were not used to generate or alter the scientific claims, experimental results, analysis, or technical content of the paper.