CLAG: Adaptive Memory Organization via Agent-Driven Clustering for Small Language Model Agents

Taeyun Roh[1] **Wonjune Jang**[2] **Junha Jung**[1,3] **Jaewoo Kang**[1,3,†]

[1]Korea University [2]Myongji University [3]AIGEN Sciences

{nrbsld, goodjungjun, kangj}@korea.ac.kr
dnjswnswkd03@mju.ac.kr

Abstract

Large language model agents heavily rely on external memory to support knowledge reuse and complex reasoning tasks. Yet most memory systems store experiences in a single global retrieval pool which can gradually dilute or corrupt stored knowledge. This problem is especially pronounced for small language models (SLMs), which are highly vulnerable to irrelevant context. We introduce **CLAG**, a **CL**ustering-based **AG**entic memory framework where an SLM agent actively organizes memory by clustering. CLAG employs an SLM-driven router to assign incoming memories to semantically coherent clusters and autonomously generates cluster-specific profiles—including topic summaries and descriptive tags—to establish each cluster as a self-contained functional unit. By performing localized evolution within these structured neighborhoods, CLAG effectively reduces cross-topic interference and enhances internal memory density. During retrieval, the framework utilizes a two-stage process that first filters relevant clusters via their profiles, thereby excluding distractors and reducing the search space. Experiments on multiple QA datasets with three SLM backbones show that CLAG consistently improves answer quality and robustness over prior memory systems for agents, remaining lightweight and efficient.

1 Introduction

AI agents are becoming the primary paradigm for deploying Large Language Models (LLMs) in real-world, long-horizon tasks (Xi et al., 2023; Zhou et al., 2023; Li et al., 2023; Wang et al., 2024). To succeed in these settings, LLMs rely on internal memory mechanisms to support knowledge reuse, multi-session coherence, and complex reasoning tasks (Wang et al., 2024; Xi et al., 2025). Conventionally, basic memory systems for AI agents adhere to a standard Retrieval Augmented Generation (RAG) (Lewis et al., 2020) paradigm. In this setup, memory functions essentially as a static repository designed to overcome fixed context window limitations. Agents sequentially record raw interaction logs, covering environmental observations, executed actions, tool output, and feedback loops during task execution (Shinn et al., 2023; Yao et al., 2023). When faced with a new state, the agent simply retrieves relevant historical snippets based on semantic similarity to ground its current input, treating memory merely as a fixed-access database of past episodes. Consequently, this static approach prevents the agent from learning from subsequent feedback or refining outdated information based on new experiences (Zhong et al., 2023; Wang et al., 2023a). Moving beyond static storage, frameworks demonstrating *agentic memory* (Xu et al., 2025; Kang et al., 2025; Yan et al., 2025) integrate active agents into memory management to enable continuous self-evolution. However, as illustrated in Figure 1, these systems typically operate within a **single global memory pool**, where both evolution and retrieval processes are conducted across the entire unstructured storage. Consequently, as a memory buffer grows, retrieval becomes vulnerable to two coupled issues: (i) the search space expands, increasing the probability of retrieving semantically plausible but task-irrelevant memories, and (ii) memory evolution mechanisms are exposed to topic-mixed neighborhoods, which can misguide updates and gradually degrade the memory store. These problems are especially salient for small language models (SLMs), which are highly vulnerable to irrelevant context (Mallen et al., 2023; Shi et al., 2023; Yoran et al., 2023; Lu et al., 2024).

To address these limitations, we propose **CLAG**, a **CL**ustering-based **AG**entic memory framework

[†]Corresponding author.

Our code is available at https://github.com/dmis-lab/CLAG

Findings of the Association for Computational Linguistics: ACL 2026, pages 16707–16726
July 2-7, 2026

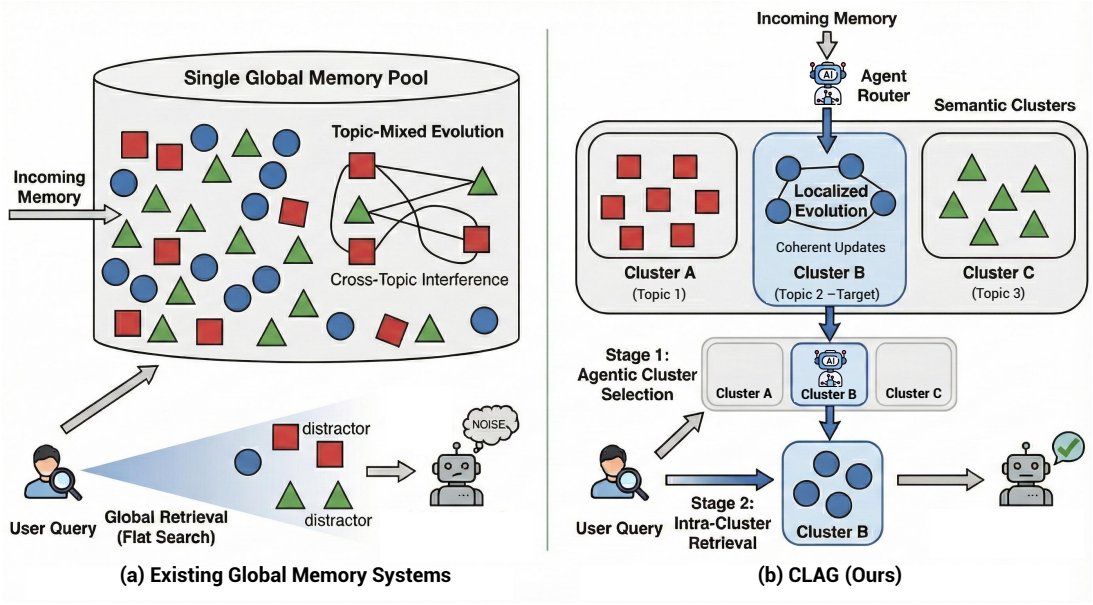

Figure 1: Conceptual comparison between existing global memory systems and CLAG. **(Left)** Traditional approaches manage memories in a single global pool, where topic-mixed updates and retrieval lead to high interference and noise accumulation. **(Right)** CLAG employs agent-driven clustering to organize memories into semantically coherent neighborhoods. By confining evolution and retrieval to these local clusters, our framework significantly reduces cross-topic interference and enhances memory stability.

that imposes lightweight structure on long-horizon memory while preserving the self-evolving nature of modern agentic systems. The core idea is to treat clustering as an *agent-controlled operation* rather than a static offline preprocessing step. Upon memory write, the agent assigns each new memory to a semantically coherent cluster through SLM-agent guided routing and maintains cluster-specific profiles. This process establishes topic-consistent neighborhoods where memory evolution is conducted locally. This design aligns with cognitive principles suggesting that new information should refine relevant schemas without perturbing unrelated memory structures (Tse et al., 2007; Kumaran et al., 2016; Gilboa and Marlatte, 2017). By confining evolution to these semantic neighborhoods, CLAG prevents noisy updates from propagating across the global store, allowing each cluster to act as a self-organizing unit that continuously enhances its internal coherence through local interactions.

At inference time, CLAG adopts a **two-stage retrieval strategy**. Given a query, the agent first selects a small set of relevant clusters using centroid-based filtering and an agentic cluster-selection module, and then retrieves fine-grained memories only within the selected clusters. This hierarchical design reduces the retrieval scope and excludes semantically plausible but task-irrelevant memories that are more likely to appear under global retrieval. Across multiple QA benchmarks and three SLM backbones, we find that this structured control loop yields consistent improvements in answer quality and robustness compared to prior agent memory baselines, while remaining lightweight in both computation and storage. These gains align with our analysis: clustering produces cleaner semantic neighborhoods for evolution and retrieval, and the two-stage retrieval pipeline mitigates the distractor exposure inherent in global retrieval.

Our contributions are threefold:

- We introduce an **agent-driven clustering** mechanism that organizes memories into semantically coherent groups via SLM-agent-based routing, ensuring robust long-horizon organization.
- We propose **localized memory evolution** that updates and consolidates memories within topic-consistent neighborhoods to stabilize long-term memory quality and mitigate cross-topic interference.
- We develop a **two-stage** cluster-aware retrieval scheme that improves robustness for limited-capacity models by reducing the search space, suppressing distractors, and lowering retrieval noise.

2 Related Work

2.1 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) (Lewis et al., 2020) extends the context window of LLMs by coupling a parametric model with a non-parametric external knowledge base. In agentic settings, the knowledge base is replaced by a memory serving as a repository for interaction histories with the environment. Typically, agents rely on retrieval over these accumulated records to inform current decision-making (Park et al., 2023; Zhang et al., 2025; Yao et al., 2023).

While variants such as hierarchical indexing (Sarthi et al., 2024; Rezazadeh et al., 2024; Edge et al., 2025; Wang et al., 2025a; Xi et al., 2023) or query rewriting (Ma et al., 2023; Wang et al., 2023b, 2025b) improve coverage, most standard RAG systems operate on a *read-only* basis over a static index. They typically process queries as independent, stateless events, lacking the mechanism to dynamically consolidate or restructure memory based on the agent's continuous experience (Packer et al., 2024).

2.2 Memory system for LLM agents

As LLMs are increasingly deployed as AI agents in long-horizon settings, retrieval has been repurposed into memory for LLM agents: storing interaction traces (observations, actions, tool outputs, feedback) and retrieving relevant past snippets to guide current decisions (Yao et al., 2023; Zhang et al., 2025). Early agent memories largely followed a static RAG-style design, treating memory as an append-only repository queried by global similarity search, which limits learning from feedback and consolidation over time. Recent agentic and evolving memory systems (e.g., A-mem (Xu et al., 2025), MemoryOS (Kang et al., 2025), GAM (Yan et al., 2025)) introduce maintenance operations such as reflection, compression, and rewriting, but typically still rely on a single global memory pool. In contrast, we reduce reliance on global search by *agentically* clustering memories online and routing new entries to an appropriate cluster, conditioning retrieval and memory operations on cluster-level structure.

3 CLAG

As illustrated in Figure 2, the CLAG framework comprises three core components designed to structure and manage long-horizon memory: (1) **Agentic Routing**, which assigns incoming memories to semantically coherent clusters; (2) **Localized Evolution**, which consolidates information within specific clusters to maintain topic consistency; and (3) **Cluster-Aware Two-Stage Retrieval**, which filters irrelevant clusters to suppress noise during inference. All agentic decisions in CLAG (routing, evolution, and cluster selection) are produced by the same backbone SLM, invoked multiple times with role-specific prompts (router/evolver/selector), rather than separate learned models.

3.1 Memory Structuring

Inspired by recent agentic memory frameworks such as A-mem (Xu et al., 2025), we represent an agent's long-term memory as a collection of structured memory notes that encode both the raw interaction trace and SLM-agent generated semantic annotations. Formally, let $\mathcal{M} = \{m_1, m_2, \ldots, m_N\}$ denote the set of all memory notes. Each note m_i is represented as

$$m_i = \{O_i, C_i, t_i, K_i, G_i, X_i, L_i\} \quad (1)$$

where O_i is the original interaction content, C_i is the cluster that the memory belongs to, t_i is the timestamp, K_i is a set of SLM-agent generated keywords that summarize salient concepts, G_i is a set of SLM-agent generated tags for additional information, X_i is an SLM-generated contextual description that captures higher-level semantics, and L_i stores links to other notes that are semantically related.

3.2 Memory Routing

We employ an SLM agent to route each incoming memory to the most semantically relevant cluster. The procedure detailed in Algorithm 1 operates in two phases based on the number of processed memories $N_{processed}$. During the initial "cold-start" phase (when $N_{processed} < n$), memories are buffered until sufficient data is collected to establish the initial cluster structure $\mathcal{C}$ via the InitializeClusters routine. Once initialized, subsequent memories undergo a routing process. First, a coarse filtering step identifies the top-k candidate clusters ($\mathcal{C}_{topK}$) closest to the new memory based on vector distance. Given the candidate clusters, the final selection is delegated to the SLM agent, which reviews the semantic profiles of these candidates to select the most appropriate cluster C_{routed}. Finally, a cosine similarity check against a threshold τ determines if the memory is routed to C_{routed}

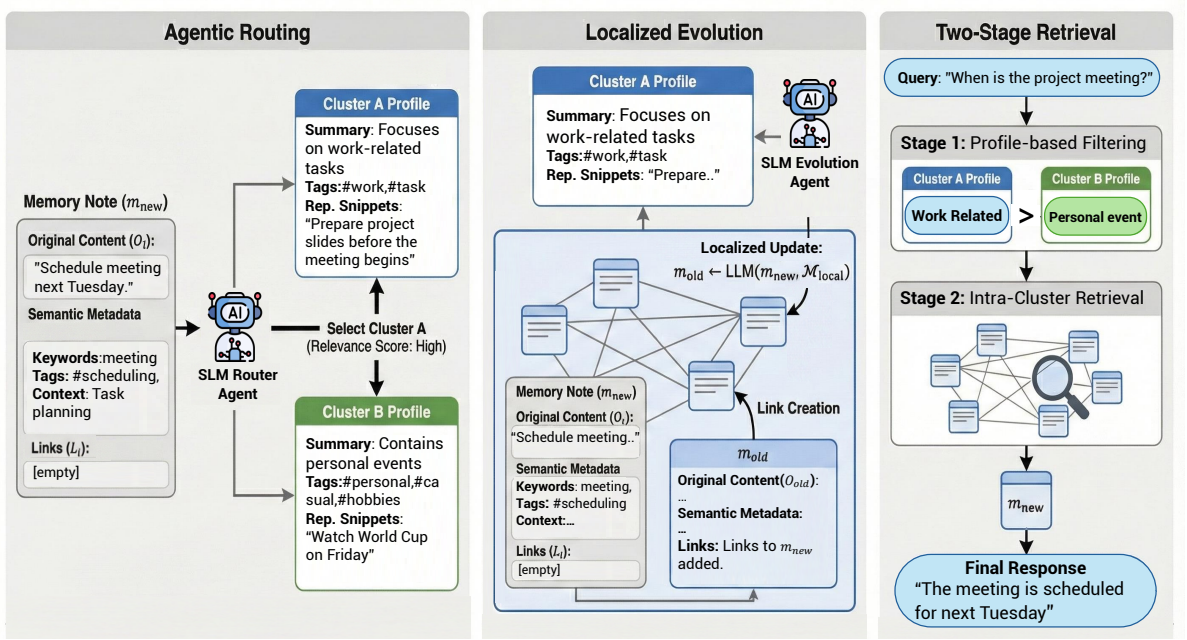

Figure 2: **Overview of the proposed CLAG framework. Left: Agentic Routing.** An SLM router assigns each incoming memory note m_{new} to the most relevant cluster using semantic metadata, and updates the corresponding cluster profile $\mathcal{P}$. **Middle: Localized Evolution.** An evolution agent performs consolidation (e.g., linking, rewriting, strengthening) *within the routed cluster* to maintain topic-consistent neighborhoods and reduce cross-topic interference. **Right: Two-Stage Retrieval.** Given a query, CLAG first filters clusters using profile-based selection (Stage 1), then retrieves fine-grained memories only inside the selected clusters (Stage 2), reducing the effective search space and suppressing retrieval noise.

or if it represents novel information requiring the creation of a new cluster C_{new}.

To ensure scalability and prevent semantic drift, we employ an adaptive clustering mechanism (Algorithm 3). When a cluster C_{target} exceeds a size threshold τ_{split}, it is dynamically bisected into two sub-clusters via K-Means. This on-the-fly refinement mitigates over-saturation, keeping the memory structure adaptive to the agent's evolving experiences.

3.3 Localized Evolution within Clusters

A distinct advantage of CLAG is that both link generation and memory evolution operate strictly within the assigned cluster C_{routed}. This localized strategy not only ensures that connections are formed within a coherent semantic context but also reduces computational overhead by precluding global pairwise comparisons.

Upon the assignment of a newly routed memory m_{new} to C_{routed}, CLAG identifies the top-k most relevant neighbors within the cluster. We denote this subset of semantically aligned memories as $\mathcal{M}_{\text{local}}$, defined via cosine similarity:

$$\mathcal{M}_{\text{local}} = \{m_j \in C_{\text{routed}} \mid \text{rank}(s_{\text{new},j}) \leq k\} \quad (2)$$

where $s_{\text{new},j}$ is the cosine similarity between $\mathbf{m}_{\text{new}}$ and $\mathbf{m}_j$.

Intra-Cluster Linking CLAG establishes explicit connections between the new memory and the retrieved candidates. Since $\mathcal{M}_{\text{local}}$ is constructed via cluster routing and similarity ranking, it inherently contains memories that are semantically proximate yet often difficult to distinguish by vector similarity alone. To address this, SLM agent analyzes the fine-grained relationships (e.g., causality, temporal sequence) between m_{new} and $\mathcal{M}_{\text{local}}$ to generate a set of links L_{new}:

$$L_{\text{new}} \leftarrow \text{SLM}(m_{\text{new}} \parallel \mathcal{M}_{\text{local}}) \quad (3)$$

Localized Evolution Following link generation, the system creates a feedback loop to refine existing knowledge. For each neighbor m_j in $\mathcal{M}_{\text{local}}$, the system determines if its attributes require an update to reflect the new information:

$$m_j^* \leftarrow \text{SLM}(m_{\text{new}} \parallel \mathcal{M}_{\text{local}} \setminus \{m_j\} \parallel m_j) \quad (4)$$

Algorithm 1: Agentic Routing for New Memory

Input: New memory data m_{new}, Current set of clusters $\mathcal{C}$, Count of processed memories $N_{processed}$
Parameters: Initialization size n, Routing top-k k, Similarity threshold τ, SLM client SLM

if $N_{processed} < n$ **then**
  Add m_{new} to initialization buffer $\mathcal{B}$;
  if $|\mathcal{B}| == n$ **then**
    $\mathcal{C} \leftarrow$ InitializeClusters($\mathcal{B}$);
else
  Identify set $\mathcal{C}_{topK}$ containing k closest clusters;
  $C_{routed} \leftarrow SLM$.SelectCluster($m_{new}, \mathcal{C}_{topK}$);
  $sim =$ CosineSimilarity($\mathbf{m}_{new}, C_{routed}$);
  if $sim < \tau$ **then**
    Create new cluster C_{new} initialized with m_{new};
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_{new}\}$;
  else
    Add m_{new} to C_{routed};
$N_{processed} \leftarrow N_{processed} + 1$;

The evolved memory m_j^* then replaces m_j within the cluster. Concurrently, the cluster profile is updated to maintain alignment with the refined semantic state of the cluster. This localized evolution mimics human cognitive processes, where new experiences primarily reshape our understanding of related concepts (i.e., the cluster) without disrupting unrelated knowledge domains (Tse et al., 2007; Kumaran et al., 2016; Gilboa and Marlatte, 2017). Consequently, each cluster functions as a self-refining module that iteratively enhances its semantic density and internal coherence through localized updates.

3.4 Cluster-Aware Two-Stage Retrieval

At inference time, CLAG replaces flat global retrieval with a cluster-aware two-stage retrieval pipeline. Given a user query q, the system initially identifies a candidate subset of relevant clusters and subsequently performs retrieval only within those clusters.

Stage 1: Agentic Cluster Selection. The query and its tags are concatenated and embedded into a vector $\mathbf{e}_q$. Based on the distance to cluster centroids $\{\mathbf{c}_k\}$, a candidate set $\mathcal{C}_{\text{cand}}$ of the K closest clusters is identified. Instead of a deterministic top-K selection, the final decision is delegated to the SLM agent. By evaluating cluster profiles against the query, agent returns a variable-size subset $\mathcal{C}_{\text{selected}} \subseteq \mathcal{C}_{\text{cand}}$, constraining the search space and suppressing irrelevant noise, which mitigates vulnerability to distractors.

Stage 2: Intra-Cluster Retrieval. Given the selected cluster set $\mathcal{C}_{\text{selected}}$, CLAG performs retrieval only within the union of their members $\mathcal{M}_{\text{cand}} = \bigcup_{C \in \mathcal{C}_{\text{selected}}} C$. By restricting retrieval to cluster-local neighborhoods, CLAG reduces the effective search space and suppresses semantically plausible but task-irrelevant memories that commonly arise under global similarity search. This hierarchical procedure is particularly beneficial for SLM agents: the agentic cluster-selection step provides a high-level semantic filter, improving robustness to retrieval noise.

4 Experiment

4.1 Datasets

We evaluate CLAG on three benchmarks. We employ **LoCoMo** (Maharana et al., 2024) and **HotpotQA** (Yang et al., 2018), which are widely adopted benchmarks for evaluating memory systems for LLM agents, to assess conversational generalization and reasoning under noise (Hu et al., 2025). Additionally, to evaluate domain adaptability, we utilize **BioASQ** (Tsatsaronis et al., 2015), which we adapted into a HotpotQA-style format to simulate noisy retrieval in specialized domains. Detailed dataset construction protocols are provided in Appendix A.

4.2 Baselines

We compare our framework against a standard **RAG** (Lewis et al., 2020) baseline and representative agentic memory architectures including **A-mem** (Xu et al., 2025), **MemoryOS** (Kang et al., 2025), and **GAM** (Yan et al., 2025); detailed implementation settings are provided in Appendix B.

4.3 Implementation Details

In our experiments, we employ Llama-3.2-1B-Instruct (Grattafiori et al., 2024), Qwen3-0.6B (Yang et al., 2025), and DeepSeek-R1-Distill-

Model	Method	LoCoMo		HotpotQA		BioASQ	
		F1	BLEU-1	F1	BLEU-1	F1	BLEU-1
Qwen3-0.6B	RAG	12.9	10.39	11.75	11.17	2.4	1.71
	A-mem	14.29	11.8	12.04	10.65	3.61	2.83
	GAM	16.05	13.24	7.81	6.69	3.40	3.37
	MemoryOS	4.30	3.24	9.02	7.34	3.12	1.29
	CLAG (Ours)	**20.99**	**17.88**	**15.50**	**14.33**	**22.01**	**17.23**
Llama3.2-1B Instruct	RAG	18.04	15.29	9.46	8.19	5.12	3.42
	A-mem	14.80	12.20	11.19	10.01	5.38	4.95
	GAM	**22.63**	**19.85**	13.85	**12.84**	6.52	4.71
	MemoryOS	9.67	6.03	6.43	4.76	3.55	2.52
	CLAG (Ours)	21.05	18.16	**14.20**	12.30	**10.16**	**8.08**
DeepSeek-R1 Distill-Qwen 1.5B	RAG	11.54	9.13	5.54	4.41	2.81	2.12
	A-mem	12.50	9.83	6.06	4.81	4.45	3.07
	GAM	12.34	9.27	6.45	4.55	2.78	2.19
	MemoryOS	6.41	4.76	4.51	2.84	2.93	1.63
	CLAG (Ours)	**15.46**	**12.6**	**9.12**	**6.63**	**6.18**	**4.42**

Table 1: Overall results on LoCoMo, HotpotQA, and BioASQ across three backbone models. Best results are marked in **bold**, 2nd-best results are marked in underline.

Method	Retrieval (ms)	End-to-End (ms)
RAG	17.80	289.60
A-mem	18.54	408.68
GAM	8303.41	17934.32
MemoryOS	220.79	968.75
CLAG (Ours)	142.43	514.14

Table 2: Latency comparison on Qwen3-0.6B backbone. All measurements were conducted on a single NVIDIA A100 GPU. Values represent the average time in milliseconds (ms).

Qwen-1.5B (Guo et al., 2025) as the backbone models for both CLAG and all baselines. For semantic embedding and retrieval tasks, we utilize the MiniLM-L6-v2 model. Further details regarding hyperparameter configurations and implementation specifics are provided in Appendix D.

5 Results

5.1 Main Results Analysis

Backbone-wise trends and robustness Table 1 shows that CLAG yields consistent gains across backbones and datasets, with particularly strong improvements on knowledge-intensive and domain-shift settings. For Qwen3-0.6B, CLAG achieves the best F1/BLEU-1 on all three benchmarks, indicating that cluster-based routing reliably suppresses distractor retrieval and improves faithfulness under long-context noise. For DeepSeek-R1-Distill-Qwen-1.5B, CLAG again provides the strongest overall results, suggesting that the proposed routing mechanism remains effective even when the backbone is changed.

Accuracy–efficiency trade-off Beyond accuracy, CLAG is designed to avoid the high computational overhead typical of multi-step agentic memory pipelines. As quantified in Table 2, on LoCoMo with the Qwen3-0.6B backbone, CLAG achieves orders-of-magnitude lower end-to-end latency than GAM, while maintaining strong accuracy. Therefore, even in regimes where CLAG and GAM are close in LoCoMo accuracy (e.g., under Llama3.2-1B-Instruct), CLAG provides a markedly better operating point in practice, achieving a decisive balance between model performance and computational cost.

5.2 Performance Analysis across Question Categories

We analyze CLAG's performance across the subcategories of the LoCoMo dataset (Table 3). In complex reasoning tasks (*Temporal* and *Multi-Hop*), CLAG delivers consistent gains, indicating that cluster-level organization helps preserve long-range dependencies while reducing distrac-

Model	Method	Multi Hop		Temporal		Open Domain		Single-Hop		Adversarial	
		F1	BLEU-1	F1	BLEU-1	F1	BLEU-1	F1	BLEU-1	F1	BLEU-1
Qwen3-0.6B	RAG	6.11	6.04	8.34	7.41	9.07	7.92	8.09	5.52	30.35	24.90
	A-mem	6.80	6.34	2.34	2.37	9.97	7.58	8.16	6.00	40.11	33.85
	GAM	8.46	7.08	6.50	6.78	**10.92**	**8.45**	9.41	6.92	41.25	34.60
	MemoryOS	2.29	1.99	7.19	5.60	8.93	7.68	3.58	2.53	3.85	2.72
	CLAG	**10.08**	**8.80**	**11.53**	**9.39**	8.68	6.57	**14.10**	**11.07**	**50.34**	**45.00**
Llama3.2-1B Instruct	RAG	5.78	6.56	2.64	2.54	12.13	9.34	8.73	5.95	55.71	48.88
	A-mem	4.91	4.19	2.65	2.76	8.02	6.07	5.92	3.68	47.87	41.51
	GAM	10.88	8.94	5.81	3.45	**16.06**	**12.62**	11.48	8.76	**63.30**	**59.75**
	MemoryOS	8.97	6.64	**10.34**	**6.21**	9.44	6.66	10.36	6.14	8.40	5.18
	CLAG	**11.24**	**10.12**	9.00	6.05	9.38	7.31	**11.52**	**8.77**	56.41	52.02
DeepSeek-R1 Distill-Qwen-1.5B	RAG	4.62	4.40	2.95	2.15	7.49	6.77	6.42	4.31	32.64	26.77
	A-mem	5.73	5.70	2.21	1.95	10.55	8.84	7.78	5.27	33.52	26.92
	GAM	6.87	6.62	5.53	4.96	9.10	7.94	8.21	5.39	29.65	21.65
	MemoryOS	4.74	5.20	6.51	5.26	8.02	6.19	6.64	4.36	6.63	4.59
	CLAG	**6.99**	**7.02**	**8.92**	**8.02**	**11.70**	**10.78**	**8.58**	**5.87**	**39.33**	**32.53**

Table 3: Detailed experimental results on the LoCoMo dataset across five categories. Results are reported in F1 and BLEU-1 (%). Best results are marked in **bold**, 2nd-best results are marked in underline.

tion from irrelevant memory. CLAG achieves the best Multi-Hop F1 across all backbones and improves Temporal performance for two models (e.g., Qwen3-0.6B: 11.53; DeepSeek-R1-Distill-Qwen-1.5B: 8.92), with BLEU-1 showing a similar trend.

For *Adversarial* questions, CLAG yields strong improvements, supporting our claim that localized retrieval mitigates distractor-heavy failure modes. In contrast, *Open Domain* results are more mixed: methods with broader retrieval (e.g., GAM) can outperform CLAG for some backbones, suggesting a trade-off between localization and coverage. Overall, this category-wise breakdown highlights where CLAG is most effective (hard reasoning and robustness) and motivates adaptive broadening as a potential extension for open-domain queries.

5.3 Retrieval Performance and Memory Quality Analysis

To investigate the sources of our performance gains, we analyze the retrieval quality across three benchmarks with Qwen3-0.6B. Table 4 presents the comparative results on evidence retrieval (E-F1) and ranking metrics (Recall@K, nDCG@K). We exclude MemoryOS and GAM from this specific retrieval-quality analysis because they do not adhere to a standard K-bounded raw-evidence retrieval interface. Both systems yield a dynamic, variable number of retrieved items driven by architectural design, rendering them incompatible with standard K-bounded evaluation protocols. Detailed results of budget-aware retrieval quality experiments, including these two baselines, are provided in Appendix I.

Structural Advantage in Retrieval On the LoCoMo and BioASQ benchmarks, CLAG demonstrates the clear structural advantage of *Agentic Clustering* and *Two-Stage Retrieval*. In LoCoMo, which involves long conversational histories, CLAG achieves an E-F1 of 2.07, significantly outperforming A-mem (1.17) and RAG (1.12). This indicates that the agent-driven routing effectively filters out irrelevant interaction logs that often confuse global retrievers. The advantage is even more pronounced in the biomedical domain (BioASQ), where CLAG exhibits a decisive lead with an E-F1 of 25.11, exceeding baselines (RAG: 2.29, A-mem: 2.20) by an order of magnitude. In domains laden with dense terminology, global similarity search often struggles to distinguish relevant context from distractor passages. By leveraging the agentic router to narrow the search space to semantically coherent clusters, CLAG effectively filters out domain-specific noise, ensuring high-fidelity retrieval.

Localized Evolution Gains In the HotpotQA benchmark, CLAG achieves retrieval performance

Dataset	Method	E-Prec	E-Recall	E-F1	R@5	R@10	nDCG@10
LoCoMo	RAG	0.66	4.83	1.12	3.12	4.83	3.26
	A-mem	0.68	5.50	1.17	3.86	5.50	3.62
	CLAG (Ours)	**1.20**	**9.74**	**2.07**	**7.18**	**9.74**	**6.89**
HotpotQA	RAG	9.75	**14.71**	**11.44**	**10.71**	**14.71**	23.71
	A-mem	8.05	12.63	9.60	8.38	12.63	22.56
	CLAG (Ours)	**9.86**	14.20	11.17	9.94	14.20	**23.98**
BioASQ	RAG	4.60	1.65	2.29	1.48	1.65	20.19
	A-mem	4.40	1.59	2.20	1.48	1.59	21.27
	CLAG (Ours)	**33.35**	**32.64**	**25.11**	**25.90**	**32.64**	**56.17**

Table 4: Retrieval performance analysis across three datasets (Backbone: Qwen3-0.6B). Metrics include Evidence Precision (E-Prec), Recall (E-Recall), F1 (E-F1), Recall at k (R@k), and nDCG@10. Best results are marked in **bold**, 2nd-best results are marked in underline.

Clustering Strategy	F1	BLEU-1
Cosine-based Clustering	14.78	12.53
K-Means Clustering	15.64	13.36
CLAG (Ours)	**22.01**	**17.23**

Table 5: Impact of different clustering methods on BioASQ (Qwen3-0.6B).

comparable to the RAG baseline (E-F1 11.17 vs. 11.44) and slightly better than A-mem (9.60), while securing the highest ranking score (nDCG@10 23.98). Despite the similarity in raw retrieval metrics, CLAG demonstrates superior final answer quality (as shown in Table 1). This discrepancy highlights the critical contribution of *Localized Evolution*. CLAG continuously consolidates and rewrites memories within topic-consistent neighborhoods. This process increases the information density of each memory note. Consequently, even when the retrieval recall is on par with baselines, the retrieved content in CLAG provides a synthesized context, enabling the agent to generate accurate reasoning steps. Detailed ablation study quantifying the specific contributions of localized evolution and two-stage retrieval is provided in Appendix C.

5.4 Agentic vs. Geometric Clustering

Motivated by the retrieval-quality findings in Section 5.3, we further isolate the contribution of *Agentic Clustering*. In particular, the retrieval analysis reveals that CLAG delivers the largest gain in evidence retrieval on BioASQ (E-F1: 25.11), where dense biomedical terminology and frequent lexical overlap make global similarity search prone to retrieving high-similarity yet irrelevant passages. This suggests that the primary challenge in specialized domains lies in the pre-retrieval organization of the search space, not in the capability of the retriever itself.

To directly test this hypothesis, we replace CLAG's agentic clustering with representative non-agentic clustering strategies while keeping the rest of the pipeline unchanged, and evaluate on BioASQ. Specifically, we compare our approach against two geometric baselines: (1) **Cosine-based Clustering**, which greedily assigns each incoming memory to the cluster centroid yielding the highest cosine similarity, and (2) **K-Means Clustering**, which partitions the memory space based purely on geometric proximity in the embedding space.

As shown in Table 5, non-agentic clustering provides limited gains. In contrast, **CLAG** achieves a substantial leap in final answer quality (F1: 22.01, BLEU-1: 17.23), far exceeding the best non-agentic alternative (K-Means: 15.64/13.36).

These results indicate that simply grouping memories by geometric similarity is insufficient in domains with heavy jargon and ambiguous surface cues. Combined with the evidence from Section 5.3, this controlled ablation supports the view that *Agentic Clustering* is a central driver of CLAG's robustness in specialized retrieval settings.

5.5 Effect of Model Scale

Our primary focus is on resource-constrained SLMs ($<$ 2B parameters), where retrieval noise and irrelevant context more easily degrade reasoning. This regime is especially important for on-device and cost-sensitive agentic applications, as smaller backbones are generally less robust to dis-

Method	LoCoMo		HotpotQA		BioASQ	
	F1	BLEU-1	F1	BLEU-1	F1	BLEU-1
RAG	19.06	15.93	23.66	21.33	22.21	20.19
A-mem	21.44	**18.35**	22.73	20.94	18.59	15.86
GAM	19.55	17.26	22.09	19.30	22.96	19.42
MemoryOS	19.60	15.89	24.64	22.37	**28.83**	22.48
CLAG (Ours)	**21.50**	17.94	**25.40**	**23.53**	25.40	**23.54**

Table 6: Results on the stronger backbone Qwen3-8B. CLAG remains competitive as model scale increases, achieving the best results on LoCoMo and HotpotQA and the best BLEU-1 on BioASQ. Best results are marked in **bold**, and second-best results are marked in underline.

tractors than larger models.

To test whether CLAG remains effective beyond this setting, we additionally evaluate Qwen3-8B. Table 6 shows that CLAG remains competitive, achieving the best results on LoCoMo and HotpotQA and comparable performance on BioASQ.

We further observe that the relative gap between CLAG and the baselines narrows as model scale increases. We attribute this trend to the stronger intrinsic robustness of larger backbones, which are better able to tolerate noisy evidence (Shi et al., 2023). In contrast, CLAG's retrieval filtering is particularly valuable for SLMs, where suppressing distractors is essential for stable agentic reasoning.

6 Conclusion

We introduce **CLAG**, a framework that structures long-horizon agentic memory through agent-driven clustering and localized evolution. By shifting from a static global pool to dynamic semantic neighborhoods, **CLAG** ensures that memory updates refine relevant schemas while minimizing cross-topic interference. Furthermore, our cluster-aware two-stage retrieval mechanism effectively suppresses noise, addressing the critical vulnerability of SLMs to distractor-heavy contexts. Experiments confirm that this unified approach significantly improves robustness and scalability, offering a lightweight yet high-performing memory solution tailored for limited-capacity agents. Overall, CLAG is a practical memory layer for limited-capacity agents, improving robustness without introducing substantial runtime latency. We also anticipate extensions to safer deployment, including access control and retention policies for persistent memory.

Limitations

A primary technical limitation of CLAG is its reliance on prompt-based cluster profile generation and agentic behavior. Consequently, performance can be sensitive to specific prompt designs and variations in the underlying language model versions. Furthermore, the quality of routing decisions may vary under significant distribution shifts, such as newly emerging topics or drastic changes in writing style, which were not systematically evaluated in this study.

From a broader perspective, the organization and persistent storage of user memories raise important privacy and data governance questions. While this work focuses on the retrieval and organization mechanisms rather than deployment policies, practical application requires robust safeguards. Real-world deployment must incorporate strict access controls, clear retention and deletion policies, and transparent user consent mechanisms, especially when handling sensitive or personally identifiable information.

Acknowledgments

We thank Yein Park, Hyeon Hwang, Taewhoo Lee, Minju Song and Jonghoon Lee for their insightful feedback and discussions. This research was supported by (1) the National Research Foundation of Korea (NRF-2023R1A2C3004176), (2) the Ministry of Health & Welfare, Republic of Korea (HR20C002103), (3) ICT Creative Consilience Program through the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (IITP-2026-RS-2020-II201819), (4) the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT and MOE) (No. RS-2025-16652968), and (5) the Seoul National University Hospital with support from the Ministry of Science and ICT (RS-2023-00262002).

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From local to global: A graph rag approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

Asaf Gilboa and Hannah Marlatte. 2017. Neurobiology of schemas and schema-mediated memory. *Trends in cognitive sciences*, 21(8):618–631.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, and 28 others. 2025. Memory in the age of ai agents. *Preprint*, arXiv:2512.13564.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. In *ACM Transactions on Information Systems*, pages 422–446.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. *arXiv preprint arXiv:2506.06326*.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. 2016. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534.

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

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D Lane, and Mengwei Xu. 2024. Small language models: Survey, measurements, and insights. *arXiv preprint arXiv:2409.15790*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

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

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. *Preprint*, arXiv:2310.08560.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

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*, pages 1–22.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. 2024. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for llms. *arXiv preprint arXiv:2410.14052*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *Preprint*, arXiv:2401.18059.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can

be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, and 1 others. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):138.

Dorothy Tse, Rosamund F Langston, Masaki Kakeyama, Ingrid Bethus, Patrick A Spooner, Emma R Wood, Menno P Witter, and Richard GM Morris. 2007. Schemas and memory consolidation. *Science*, 316(5821):76–82.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2025a. Scm: Enhancing large language model with self-controlled memory framework. *Preprint*, arXiv:2304.13343.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *Preprint*, arXiv:2305.16291.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Liang Wang, Nan Yang, and Furu Wei. 2023b. Query2doc: Query expansion with large language models. *arXiv preprint arXiv:2303.07678*.

Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2025b. Speculative rag: Enhancing retrieval augmented generation through drafting. *Preprint*, arXiv:2407.08223.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, and 10 others. 2023. The rise and potential of large language model based agents: A survey. *Preprint*, arXiv:2309.07864.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

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

BY Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. 2025. General agentic memory via deep research. *arXiv preprint arXiv:2511.18423*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

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*, pages 2369–2380.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and 1 others. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations (ICLR)*.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2023. Memorybank: Enhancing large language models with long-term memory. *Preprint*, arXiv:2305.10250.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

A Dataset Details

We evaluate our method on three benchmarks—LoCoMo, HotpotQA, and BioASQ—chosen to cover conversational long-term memory, reasoning under noisy contexts, and domain-specific adaptation.

LoCoMo We use LoCoMo (Maharana et al., 2024), one of the longest and most comprehensive benchmarks for *memory for agents* in long-context dialogue settings. Unlike prior datasets that are typically limited to five chat sessions, LoCoMo features very long-term conversations spanning up to 35 sessions, with an average of 300 turns and 9K tokens per dialogue. These conversations are generated via a human-machine pipeline grounded on temporal event graphs and rigorously verified by human annotators to ensure long-range consistency. This dataset is specifically designed to challenge models in comprehending long-range temporal and causal dynamics, making it a rigorous testbed for evaluating the robustness of our memory system.

HotpotQA We build on HotpotQA (Yang et al., 2018) under the distractor construction used in MemAgent and GAM (Yu et al., 2025; Yan et al., 2025), where each query is paired with gold supporting documents and additional irrelevant passages sampled from the same corpus to form a long noisy context. For reproducibility and fair comparison in the small-agent regime, we use a fixed-passage (bounded-context) setting with 20 passages per question (gold evidence + distractors). In our HotpotQA-20 construction, concatenating the chunked passages results in approximately 33–39k tokens per query (median $\approx$ 35k; 25–75%: 34–37k), which is substantially shorter than the 56K/224K/448K variants in MemAgent and avoids small-model context collapse.

BioASQ To evaluate domain adaptability, we adapt BioASQ Task 10b (Tsatsaronis et al., 2015) using the same distractor construction protocol as HotpotQA. We filtered the dataset to retain only factoid and list types compatible with standard QA metrics (e.g., F1, BLEU-1). Specifically, we randomly sampled 200 questions (140 factoid and 60 list items) from this subset to form our evaluation benchmark. For reproducibility and fair comparison in the small-agent regime, we use a bounded-context setting with 20 passages per question (gold evidence + distractors). In our BioASQ-20 construction, concatenating these passages yields inputs of approximately 5–16k tokens (median $\approx$ 10.7k; 25–75%: 9.4–12.9k), keeping the context length within the range where small models remain stable.

B Baseline Details

B.1 Baselines Description

We provide descriptions of each baseline to clarify their memory assumptions and architectural differences from our approach.

RAG RAG (Lewis et al., 2020) augments a language model with an external retriever that fetches semantically relevant documents for each query. Following the setting used in the LoCoMo benchmark, the retrieved passages are concatenated with the query and provided to the model as additional context.

GAM General Agentic Memory (GAM) (Yan et al., 2025) is a just-in-time (JIT) memory framework that builds task-specific context at runtime instead of relying on heavily pre-compressed static memory. It uses a Memorizer to maintain lightweight cues while storing full histories in a page-store, and a Researcher to retrieve and integrate relevant information from the page-store for each query, guided by those cues. Critically, the Researcher operates through an iterative loop—planning search actions, retrieving data, and reflecting on the results—until the internal information need is fully satisfied. This dynamic termination criterion results in a variable number of retrieved contexts per query, making the system incompatible with standard fixed-K ranking evaluations.

MemoryOS MemoryOS (Kang et al., 2025) proposes an operating-system-inspired memory architecture that organizes information into short-term, mid-term, and long-term memory layers. It employs automated memory promotion, decay, and summarization strategies to manage memory persistence, enabling scalable long-term interaction without explicit agentic routing decisions. During retrieval, MemoryOS aggregates hierarchical outputs and persona information from these multiple tiers to construct the final context. Since the volume of retrieved information is determined by the system's architectural depth and the current state of the hierarchy rather than a user-defined budget, it cannot be aligned with standard K-bounded retrieval metrics.

Variant	F1	BLEU-1
CLAG (full)	22.01	17.23
w/ Global Evolution	19.55	17.25
w/ Global Retrieval	20.25	16.56

Table 7: Ablation on BioASQ (Qwen3-0.6B). Replacing localized evolution with global evolution drops F1 by 2.46, and replacing two-stage retrieval with global retrieval drops F1 by 1.76.

A-mem A-mem (Xu et al., 2025) introduces an agentic memory framework in which a dedicated memory agent dynamically decides what information to store, update, or retrieve based on task requirements. The system relies on LLM-driven reasoning for memory management, providing flexibility at the cost of increased computational overhead.

C Impact of Localized Evolution and Two-Stage Retrieval

Experimental Setup. We quantify the contribution of (i) *Localized Evolution* and (ii) *Cluster-aware Two-Stage Retrieval* using BioASQ under the Qwen3-0.6B backbone. We compare the full system (**CLAG**) against two controlled ablations: **(a) Global Evolution** which evolves memories beyond the assigned local neighborhood, and **(b) Global Retrieval** which replaces the two-stage pipeline with a flat global similarity search. We report answer quality with F1 and BLEU-1.

C.1 Localized Evolution Improves Answer Quality with Local Computation

Replacing localized evolution with global evolution reduces F1 from 22.01 to 19.55 (-2.46). This indicates that continuously consolidating memories within topic-consistent neighborhoods is a key driver of final answer quality, likely by increasing the information density and coherence of notes before retrieval and generation.

Cost-efficiency via local neighborhood updates. Localized evolution rewrites only the memories within the assigned cluster (local neighborhood), avoiding a global update over the entire memory store. Conceptually,

$$\text{Cost}_{\text{local}} \propto |M_{\text{local}}| \quad \ll \quad \text{Cost}_{\text{global}} \propto |M|. \tag{5}$$

In our runs, the clustering structure is moderate in granularity: the number of clusters is 5.6 ± 3.56 (min 3, max 13; count 10), with mean cluster size 67.99 ± 41.90 and max cluster size 139.5 ± 38.47 (min 85, max 210; count 10). These statistics support that evolution can be restricted to a bounded topic neighborhood rather than the full memory set, enabling more cost-efficient continual refinement.

C.2 Two-Stage Retrieval: Modest Pruning, Consistent Gains

Replacing two-stage retrieval with global retrieval decreases F1 from 22.01 to 20.25 (-1.76) and BLEU-1 from 17.23 to 16.56 (-0.67), showing that structuring the search space before fine-grained retrieval is beneficial.

Search-space reduction. We measure search-space reduction as

$$r \;=\; 1 - \frac{|S_{\text{searched}}|}{|S_{\text{all}}|}, \tag{6}$$

where S_{searched} denotes the set of memories examined after Stage-1 cluster selection (i.e., the union of memories within the selected clusters), and S_{all} is the full memory pool. Across 200 BioASQ queries, we observe

$$\text{mean } r = 0.0802\ (\approx 8.02\%), \quad \text{std } = 0.1729,$$
$$\min = 0.0, \quad \max = 0.8957\ (\approx 89.57\%).$$

While the average pruning is modest, the high variance indicates that for a subset of queries the two-stage mechanism substantially narrows the candidate pool. Importantly, even this conservative pruning yields consistent answer-quality improvements, suggesting that the main benefit stems from reducing domain-specific distractors and improving the relevance of the candidate set rather than aggressively shrinking it.

D Hyperparameter Details

We provide the detailed hyperparameter configurations used in our experiments to ensure reproducibility. The specific values for memory initialization, dynamic clustering, and retrieval are summarized in Table 8.

For the initialization phase in Algorithm 2, we set the buffer size $n = 100$ to accumulate sufficient data before performing the initial K-Means clustering. During the online routing process Algorithm 1, the SLM router is presented with $k_{\text{route}} = 3$ candidate clusters based on embedding similarity. We employ a similarity threshold of τ to determine whether to instantiate a new cluster and a maximum

Hyperparameter	Meaning	Value
n	Init buffer size (min memories before initial KMeans)	100
n_{init}	Initial clusters for KMeans initialization	3
τ_{split}	Cluster split threshold (split if size $> \tau_{\text{split}}$)	300
k_{route}	Routing candidate clusters shown to SLM	3
τ	New-cluster threshold (min cosine sim to accept a cluster)	0.1
k_{local}	Intra-cluster neighbors for evolution/linking	5
K_{stage1}	Candidate clusters for Stage-1 selection	3
k_{retrieve}	Final retrieval top-k memories	10

Table 8: Hyperparameters used in our CLAG implementation. Unless stated otherwise, all experiments use the values shown above.

capacity threshold of $\tau_{\text{split}} = 300$ to trigger cluster splitting (Algorithm 3). To validate the stability of our proposed framework, we conducted additional ablation studies on the impact of these hyperparameter values. The experimental results exhibit small variance across a wide range of settings, demonstrating the **robustness of the architectural structure of CLAG**. This consistency indicates a **low dependency on specific hyperparameter configurations**, confirming that the system's performance is driven by the agentic mechanism itself rather than heuristic tuning.

Impact of Initial Cluster Count (n_{init}) To assess the system's sensitivity to the initial memory structure, we evaluated performance with varying numbers of initial clusters ($n_{\text{init}} \in \{3, 5, 10\}$). As shown in Table 9, increasing n_{init} from 3 to 5 yields a slight performance gain, achieving the best F1 score of 22.71 and BLEU-1 of 19.01. Further increasing the count to 10 results in a marginal decline, yet the system maintains high stability across all metrics. While $n_{\text{init}} = 5$ yields the peak score, we selected $n_{\text{init}} = 3$ as the default to align with the intrinsic semantic dimensionality of the dataset, where we observed the agent naturally gravitates towards approximately three dominant topic clusters regardless of initialization. This suggests that CLAG's continuous evolution mechanism effectively compensates for initialization choices.

Initial Clusters (n_{init})	F1	BLEU-1
3 (Default)	20.99	17.88
5	**22.71**	**19.01**
10	21.71	18.50

Table 9: Sensitivity analysis of the initial cluster count (n_{init}) on LoCoMo (Qwen3-0.6B). The system maintains high performance across all metrics, regardless of initialization settings.

Impact of New-Cluster Threshold (τ) We further analyzed the impact of the similarity threshold τ, which governs the propensity to create new clusters. Table 10 demonstrates the relationship between τ, the resulting cluster granularity, and retrieval performance. As expected, increasing τ makes the system stricter about merging new memories into existing clusters, leading to a significant increase in the average number of clusters (from 4.7 to 47.5). We prioritized structural parsimony by selecting $\tau = 0.10$ (Default) despite the peak performance at $\tau = 0.15$; this setting prevents over-segmentation into numerous micro-clusters and minimizes management overhead while maintaining competitive accuracy. Remarkably, despite the drastic change in cluster count at higher thresholds, the retrieval performance remains robust. The F1 score peaks at $\tau = 0.15$ (22.41) and degrades only slightly even at $\tau = 0.30$ (21.94). This resilience confirms the efficacy of our *Agentic Cluster Selection* (Stage 1 retrieval): even when the memory space is fragmented into many fine-grained clusters, the agent successfully identifies the relevant neighborhoods, preventing performance collapse.

Threshold (τ)	F1	BLEU-1	Avg. # Clusters
0.10 (Default)	20.99	17.88	4.7
0.15	**22.41**	**18.93**	6.9
0.20	21.68	18.29	15.6
0.25	21.99	18.35	26.5
0.30	21.94	18.73	47.5

Table 10: Sensitivity analysis of the new-cluster threshold (τ) on LoCoMo (Qwen3-0.6B). While the number of clusters increases significantly with higher thresholds, retrieval performance remains robust.

E Case Study: Mystery Novel Inquiry

To further illustrate the advantage of CLAG, we present a qualitative case study from the LoCoMo

benchmark using the Qwen3-0.6B backbone. The query and ground-truth answer are taken verbatim from LoCoMo Sample 4.

Query. *"Which two mystery novels does Tim particularly enjoy writing about?"*

Ground Truth. *Harry Potter and Game of Thrones*

Table 11 compares model predictions for this query. Among all methods, only CLAG correctly identifies both titles. In contrast, the baseline methods either abstain or generate overly vague responses that fail to recover the exact two-item answer required by the query.

Method	Prediction	Result
CLAG (Ours)	"Harry Potter and Game of Thrones"	**Correct ✓**
RAG	"Not mentioned in the conversation."	**Wrong ✗**
A-mem	"Tim's favorite two mystery novels are not mentioned."	**Wrong ✗**
GAM	"Not mentioned in the conversation."	**Wrong ✗**
MemoryOS	"Tim writes about both fantasy and mystery novels."	**Wrong ✗**

Table 11: **Comparison of generated responses** For the query *"Which two mystery novels does Tim particularly enjoy writing about?"*, only CLAG correctly identifies the two titles mentioned in the dialogue.

E.1 Agentic Memory Organization and Pruning

The memory pool for this sample contains 680 notes spanning diverse topics, including literature, sports, and social interactions. Rather than retrieving directly from the full mixed memory pool, CLAG's agentic router first organizes notes into semantically coherent clusters and then performs Stage-1 selection to identify the most relevant region for the query.

As shown in Table 12, this step selects only the literature-related cluster and prunes the others, reducing the effective search space from 680 notes to 119 notes. This corresponds to an 82.5% reduction in candidate memories before fine-grained retrieval. By filtering out unrelated clusters in advance, CLAG is able to focus on the evidence directly tied to Tim's writing preferences and recover the correct titles.

Cluster ID	Profile	Count	Status
0	Speaker discusses how books create new worlds	119	**Selected ✓**
1	Impact of basketball on community growth	231	Pruned ✗
2	Experience of meeting teammates after a trip	330	Pruned ✗

Table 12: **Semantic clustering and pruning for the case study sample.** CLAG selects the literature-related cluster and prunes away unrelated clusters, reducing the search space from 680 to 119 notes before fine-grained retrieval.

F Algorithmic Details

In this section, we provide the detailed procedure for cluster initialization and splitting, which are core components of our agentic memory maintenance. Specifically, Algorithm 2 details the bootstrapping of the initial memory hierarchy through semantic partitioning, while Algorithm 3 governs the dynamic refinement of clusters to ensure high-granularity memory maintenance.

Algorithm 2: Initialize Clusters

Input: Initialization buffer of memories $\mathcal{B}$
Parameters: Target number of initial clusters n_{init}, SLM client SLM
Output: Initialized set of clusters $\mathcal{C}$

Extract embeddings E from memories in $\mathcal{B}$;
Partition E into n_{init} sets using KMeans: $\{M_1, M_2, \ldots, M_{n_{init}}\}$;
$\mathcal{C} \leftarrow \emptyset$;
for $i \leftarrow 1$ **to** n_{init} **do**
    Calculate centroid $\mathbf{c}_i$ of memories in M_i;
    Generate Cluster profile $profile_i \leftarrow SLM$.GenerateProfile(M_i);
    Create cluster $C_i \leftarrow$ {memories : M_i, centroid : $\mathbf{c}_i$, profile : $profile_i$};
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_i\}$;
return $\mathcal{C}$;

G Failure Analysis of Agentic Components

We analyze failure cases of the agentic router and selector in CLAG on LoCoMo with Qwen3-0.6B.

Algorithm 3: Split Cluster

Input: Target cluster ID cid, Global set of clusters $\mathcal{C}$, Memory storage $\mathcal{M}$
Parameters: Split threshold τ_{split}
Output: Updated set of clusters $\mathcal{C}$
Retrieve cluster info $C_{target} \leftarrow \mathcal{C}[cid]$;
if $|C_{target}.members| \leq \tau_{split}$ **then**
  return $\mathcal{C}$
Extract embeddings E from all memories $m \in C_{target}$.members;
Partition E into 2 sets using KMeans: $\{M_A, M_B\}$ with centroids $\mathbf{c}_A, \mathbf{c}_B$;
Define new cluster IDs id_A, id_B;
Create cluster $C_A \leftarrow \{\text{members} : M_A, \text{centroid} : \mathbf{c}_A, \text{id} : id_A\}$;
Create cluster $C_B \leftarrow \{\text{members} : M_B, \text{centroid} : \mathbf{c}_B, \text{id} : id_B\}$;
$\mathcal{C} \leftarrow \mathcal{C} \cup \{C_A, C_B\}$
Remove C_{target} from $\mathcal{C}$;
return $\mathcal{C}$;

Specifically, we examine examples where CLAG underperforms the global-retrieval baseline, i.e., $\text{EM} = 0$ and $\text{F1}_{\text{CLAG}} \leq \text{F1}_{\text{Baseline}}$.

G.1 Failure Taxonomy

We build a heuristic labeler based on query cues and retrieval diagnostics to group failures into seven categories (Table 13).

G.2 Failure Distribution

As shown in Table 14, *Entity confusion* is the most frequent failure pattern (46.2%), followed by *Temporal-slot ambiguity* (19.3%). These results indicate that improving sensitivity to fine-grained entity and temporal cues is a promising direction for future work.

H Comparison Details

In the main text, we primarily reported F1 and BLEU-1 scores to assess lexical overlap and retrieval accuracy. To provide a more comprehensive evaluation of semantic coherence and generation quality, we additionally report **BERTScore (BERT-F1)** and **METEOR** metrics in Table 15. BERT-F1 captures semantic similarity using contextual embeddings, while METEOR accounts for synonyms and stemming, often correlating better with human judgment regarding fluency.

Consistent with the main results, **CLAG** demonstrates superior performance across most settings, particularly when using the **Qwen3-0.6B** backbone.

- **Semantic Robustness:** On the **BioASQ** dataset with Qwen3-0.6B, CLAG achieves a METEOR score of **17.75**, drastically outperforming the baselines (which remain below 3.03). This suggests that CLAG's agentic clustering and cluster-aware retrieval enable the agent to generate responses that are linguistically richer and semantically more accurate, rather than merely copying keywords.
- **Consistency across Backbones:** Even with larger backbones like Llama3.2-1B-Instruct and DeepSeek-R1-Distill, CLAG maintains competitive or superior performance. For instance, on **HotpotQA** with Llama3.2-1B-Instruct, CLAG achieves the highest METEOR score (8.64), indicating better handling of multi-hop reasoning contexts compared to GAM and MemoryOS.

I Retrieval quality under controlled evidence budgets.

We further compare CLAG with GAM and MemoryOS under budget-aware retrieval-quality evaluation. Because GAM retrieves page-level evidence, its output is not directly compatible with the fixed-K raw-evidence setting used in our main comparison, and exact character-budget matching is infeasible. We therefore report the average number of retrieved characters actually passed to QA. For MemoryOS, whose final evidence length is indirectly controlled by segment selection, we raise the segment-selection threshold to 0.7 to obtain a budget comparable to CLAG, A-mem, and RAG. As shown in Table 16, CLAG does not outperform competitors by retrieving more text; rather, it consistently retrieves more relevant evidence under a controlled budget, with especially large gains on BioASQ.

J Evaluation Metric

We evaluate our system using complementary metrics that assess both question answering quality and retrieval effectiveness. For question answering, we employ lexical- and semantic-level metrics to measure answer correctness and meaning similarity, while retrieval performance is evaluated using

Category	Description
Entity confusion	Routed to a cluster anchored to the wrong entity.
Temporal-slot ambiguity	Routed to profiles lacking temporal cues.
Stale/drift	Baseline retrieves more temporally relevant evidence.
Compositional/Multi-hop miss	Query requires composition, but too few clusters are selected ($n \leq 1$).
Location ambiguity	Routed to profiles lacking spatial cues.
Numeric-slot ambiguity	Routed to profiles lacking numeric evidence.
Residual ambiguity	Remaining underspecified or overlapping cases.

Table 13: Failure taxonomy for CLAG.

Pattern	Count	%
Entity confusion	110	46.2
Temporal-slot ambiguity	46	19.3
Stale / drift	34	14.3
Compositional / Multi-hop miss	17	7.1
Residual ambiguity	13	5.5
Location ambiguity	9	3.8
Numeric-slot ambiguity	9	3.8
Total	**238**	**100.0**

Table 14: Failure distribution ($N = 238$).

evidence-level precision, recall, and ranking-based metrics.

Question Answering Evaluation

F1 Score. The F1 score (Rajpurkar et al., 2016) evaluates answer accuracy by jointly considering precision and recall. In span-based question answering, it is computed at the token level to reward partial overlap between predicted and reference answer spans.

$$\mathrm{F1} = \frac{2 \cdot Prec \cdot Recall}{Prec + Recall} \tag{7}$$

$$Prec = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \quad Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{8}$$

where TP, FP, and FN denote true positives, false positives, and false negatives at the token level.

BLEU-1. BLEU-1 (Papineni et al., 2002) measures unigram-level lexical precision between generated answers and references, while applying a brevity penalty to discourage overly short responses.

$$\text{BLEU-1} = \mathrm{BP} \cdot \exp(\log p_1) \tag{9}$$

$$\mathrm{BP} = \begin{cases} 1, & c > r \\ \exp(1 - r/c), & c \leq r \end{cases} \tag{10}$$

where c and r denote candidate and reference lengths, and p_1 is unigram precision.

METEOR. METEOR (Banerjee and Lavie, 2005) evaluates answer quality based on aligned unigrams while accounting for synonym matching and fragmentation penalties. This allows the metric to capture semantic similarity beyond exact lexical overlap.

$$\text{METEOR} = F_{\text{mean}} \cdot (1 - \text{Penalty}) \tag{11}$$

$$F_{\text{mean}} = \frac{10PR}{R + 9P}, \quad \text{Penalty} = 0.5 \left(\frac{ch}{m} \right)^3 \tag{12}$$

where P and R denote unigram precision and recall, m is the number of matched unigrams, and ch is the number of contiguous matched chunks.

BERT-F1. BERT-F1 (Zhang et al., 2020) evaluates semantic similarity between predicted and reference answers using contextualized token embeddings from BERT. It computes precision and recall based on maximum cosine similarity between tokens, and reports their harmonic mean, enabling robust evaluation even when exact lexical overlap is limited.

$$\text{BERT-F1} = \frac{2 \cdot P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \tag{13}$$

where P_{BERT} and R_{BERT} denote token-level precision and recall, computed using maximum cosine similarity between contextualized BERT embeddings of predicted and reference tokens.

Retrieval Evaluation

Evidence Precision and Recall. We evaluate retrieval quality using evidence-level precision and recall.

$$\text{E-Prec} = \frac{|\mathcal{R} \cap \mathcal{E}|}{|\mathcal{R}|}, \quad \text{E-Recall} = \frac{|\mathcal{R} \cap \mathcal{E}|}{|\mathcal{E}|} \tag{14}$$

Model	Method	LoCoMo		HotpotQA		BioASQ	
		BERT-F1	METEOR	BERT-F1	METEOR	BERT-F1	METEOR
Qwen3-0.6B	RAG	85.07	12.78	82.11	7.07	78.89	2.68
	A-mem	85.04	12.85	82.20	6.13	79.04	3.03
	GAM	84.91	14.39	82.57	6.33	77.05	2.66
	MemoryOS	84.15	2.68	**84.39**	5.02	80.65	1.11
	CLAG (Ours)	**85.45**	**18.57**	82.47	**10.10**	**83.56**	**17.75**
Llama3.2-1B Instruct	RAG	85.52	16.09	84.70	6.15	79.25	3.52
	A-mem	85.10	14.10	**85.44**	6.52	80.37	3.43
	GAM	**87.85**	17.61	77.23	8.07	73.63	5.04
	MemoryOS	84.08	7.48	82.23	4.73	80.23	2.71
	CLAG (Ours)	86.80	**18.13**	84.78	**8.64**	**80.92**	**7.14**
DeepSeek-R1 Distill-Qwen 1.5B	RAG	84.28	8.91	82.97	3.63	79.30	2.11
	A-mem	84.38	13.11	83.05	4.06	80.08	4.67
	GAM	**86.21**	11.73	**84.78**	4.85	81.10	2.53
	MemoryOS	85.09	4.99	81.82	3.96	**81.12**	2.51
	CLAG (Ours)	85.33	**14.74**	83.38	**6.03**	79.90	**4.78**

Table 15: Detailed experimental results on LoCoMo, HotpotQA, and BioASQ: BERT-F1 and METEOR scores. Best results are marked in **bold**, 2nd-best results are marked in underline.

where $\mathcal{R}$ is the set of retrieved memories, $\mathcal{E}$ is the set of gold evidence items, and $\mathcal{R} \cap \mathcal{E}$ denotes the set of retrieved memories that match at least one gold evidence item (determined by normalized substring matching).

Evidence F1. The harmonic mean of evidence precision and recall.

$$\text{E-F1} = \frac{2 \cdot \text{E-Prec} \cdot \text{E-Recall}}{\text{E-Prec} + \text{E-Recall}} \tag{15}$$

where E-Prec and E-Recall are defined in Equation (14).

Recall@K. Recall@K (Järvelin and Kekäläinen, 2002) measures the fraction of gold evidence retrieved within the top-K results.

$$\text{Recall@}K = \frac{1}{|\mathcal{E}|} \sum_{i=1}^{|\mathcal{E}|} \mathbf{1}[\text{rank}(e_i) \leq K] \tag{16}$$

where $\mathcal{E}$ is the set of gold evidence items, e_i denotes an evidence item, $\text{rank}(e_i)$ is the rank of the retrieved memory containing e_i (or ∞ if absent), K is the cutoff rank, and $\mathbf{1}[\cdot]$ denotes the indicator function.

nDCG@K. Normalized Discounted Cumulative Gain (Järvelin and Kekäläinen, 2002) evaluates ranking quality.

$$\text{nDCG@}K = \frac{\text{DCG@}K}{\text{IDCG@}K} \tag{17}$$

$$\text{DCG@}K = \sum_{i=1}^{K} \frac{\text{rel}_i}{\log_2(i+1)} \tag{18}$$

where $\text{rel}_i \in \{0, 1\}$ denotes the relevance of the item at rank i, K is the cutoff rank, and $\text{IDCG@}K$ is the ideal DCG obtained by ranking all relevant items at the top positions.

K SLM Prompts

This appendix provides the exact prompt templates used for SLM-agent based routing and cluster profiling.

K.1 Prompt for SLM Cluster Selection

This prompt is used in the 'SLM.SelectCluster' function to determine the best-fitting cluster for a new memory from a list of candidates.

Prompt template for Cluster Selection

```
You are a memory routing assistant.
A new memory has arrived:
- Content: {note.content}
- Context: {note.context}
- Tags: {note.tags}

Here are candidate clusters (pre-selected by
vector similarity) that might relate to this
memory:
{candidates_text}

Your task:
1. Analyze the topics and contexts of the
candidate clusters provided above.
```

Dataset	Method	Evidence Quality		Ranking		Chars
		E-Prec	E-F1	R@5	nDCG@10	
LoCoMo	GAM	1.97	**3.01**	**7.50**	6.93	17,198.04
	RAG	0.66	1.12	3.12	3.26	1,389.43
	A-mem	0.68	1.17	3.86	3.62	1,483.91
	MemoryOS	**3.24**	2.87	2.72	3.35	1,373.48
	CLAG	1.20	2.07	7.18	**9.74**	1,465.65
HotpotQA	GAM	14.44	6.50	5.38	15.00	1,234.70
	RAG	9.75	**11.44**	**10.71**	23.71	1,196.73
	A-mem	8.05	9.60	8.38	22.56	1,826.13
	MemoryOS	**19.10**	6.44	4.04	19.85	1,549.99
	CLAG	9.86	11.17	9.94	**23.98**	1,844.98
BioASQ	GAM	22.90	5.73	4.27	25.06	913.45
	RAG	4.60	2.29	1.48	20.19	1,564.12
	A-mem	4.40	2.20	1.48	21.21	1,318.54
	MemoryOS	28.50	9.27	6.72	29.40	1,228.24
	CLAG	**33.65**	**25.90**	**32.64**	**56.17**	1,465.65

Table 16: Retrieval-quality comparison with GAM and MemoryOS. Best results are marked in **bold**, and second-best results are marked in underline.

```
2. Select the single cluster_id that exhibits
the highest semantic relevance and thematic
alignment with the new memory.
3.  You MUST choose exactly one cluster_id
from the candidate list.

- Do NOT output any text that is not a valid
cluster_id.
Return ONLY a JSON object in this format
(this is an example):

{{ "choice": "cluster_1" }}
Where:
- cluster_1 must be replaced with one of the
actual cluster ids from the candidate list
above.
- Do not include comments or extra fields.
```

K.2 Prompt for Cluster Profile Generation

This prompt is used to generate a semantic summary and representative tags for a cluster based on its member memories. The placeholder `{samples_text}` is populated with actual text snippets from memories within the cluster.

Prompt template for Cluster Profile Generation

```
You are a memory clustering assistant.

Below are several memory snippets that belong
to the SAME cluster:

{samples_text}
Your task:
1. Write ONE short sentence summary that best
describes the main topic of this cluster.
2. Return EXACTLY 3 tags.
- Each tag must be a single word.
- Do NOT repeat the same tag.

Return ONLY a JSON object with the following
KEYS (this is a schema, not the actual
content):

{{ "summary": "...your one-sentence summary
here...",
"tags": ["tag_1", "tag_2", "tag_3"]
}}
```

K.3 Prompt for Retrieval Stage Cluster Selection

This prompt is employed during the first stage of the retrieval pipeline (Agentic Cluster Selection). The SLM evaluates the semantic relevance of candidate clusters—identified via centroid dis-

tance—against the user's query. By allowing the agent to select a variable number of clusters, this step acts as a semantic gatekeeper, filtering out irrelevant topics before fine-grained retrieval ensues.

Prompt template for Retrieval stage cluster selection

```
You are an AI memory router that selects the most relevant memory clusters for a given query.

You will be given several candidate clusters. Each cluster has:

   • cluster_id

   • one-sentence summary

   • optional tags

   • one or more representative memory examples

Your task:
1. Analyze the user query and query_tags.
2. For each candidate cluster, judge how relevant it is.
3. Decide how many clusters are actually needed. You should return between 0 and {top_n} clusters.

   • If one cluster is definitely sufficient for answering the query, return just that one.

   • Include additional clusters if they are needed for answering the query.

4. If none of the clusters are meaningfully related, return an empty list.

Return ONLY JSON with this format:
{
"selected_clusters":
[ "cluster_id_1", "cluster_id_2" ]
}

If no cluster is relevant, return:
{
"selected_clusters": []
}

User query: {query}
Query tags: {query_tags}

Candidate clusters:
{candidate_clusters_text}
```