Agentic Memory Tagging: Scalable Yet Stable Indexing for Long-Term Memory

Shiyu Zhu[1], Miao Fan[1,*], Wenjie Zhou[2], Chen Ma[3], Xiangzeng Liu[4], Haoyi Xiong[5]

[1]Beijing Institute of Graphic Communication, [2]Renmin University of China,
[3]City University of Hong Kong, [4]Xidian University, [5]Independent Scholar

***Abstract*—Long-term memory is essential for multi-session interaction in large language model (LLM) agents. However, existing memory mechanisms lack a definitive indexing structure, forcing them to rely on inefficient global similarity searches during the retrieval phase. As memory accumulates throughout the agent-user iterations, this lack of structure results in severe storage redundancy and fragmented indexes, which compromise the coherence of the retrieved context. To address these structural bottlenecks, we propose agentic memory tagging, a training-free framework that constructs and maintains a stable memory index. Rather than deferring organization to retrieval, the agent assigns a primary semantic tag as an explicit storage key to each incoming memory, deterministically placing it into a corresponding tag box. To maintain index stability, the agent also aligns synonymous tags to shared canonical keys and merges redundant entries within each box when their count exceeds a set threshold. This structure allows queries to be quickly directed to their relevant tag boxes for efficient access. This design ensures a stable indexing structure even as memory grows continuously. Experiments on the LOCOMO benchmark show that our approach significantly reduces redundancy and fragmentation, enabling more coherent and efficient memory retrieval.**

***Index Terms*—LLM agents, long-term memory, semantic tagging, memory indexing**

I. Introduction

Large language model (LLM) agents are increasingly equipped with long-term memory to maintain coherent interactions in multiple sessions and diverse topics [1]–[3]. Beyond simple recall of facts, these agents are expected to maintain consistency in the user's state, track evolving preferences, and integrate information over extended horizons. Although context windows and retrieval-augmented generation (RAG) have advanced significantly, long-horizon dialog benchmarks show that performance still lags: LLMs continue to struggle with long-range temporal dependencies and complex multi-hop reasoning [4], [5]. These findings suggest that simply scaling context length is insufficient; instead, there is a critical need for external memory mechanisms capable of intelligently organizing and reliably retrieving information over time [6]. Current memory systems predominantly follow a retrieval-centric paradigm, storing interactions as independent entries and deferring organization until retrieval via similarity or heuristics [3], [7], [8]. As memory scales, this approach inevitably leads to severe storage redundancy and index fragmentation, producing a noisy context that compromises response coherence [9]. Although structured methods such as A-Mem [13] or graph-based representations [10]–[12] improve flexibility, they often fail to resolve fragmentation when identical concepts appear in various lexical forms. In these systems, tags function merely as passive metadata [14], shifting the burden of reconciling scattered weakly connected entries to the retrieval phase.

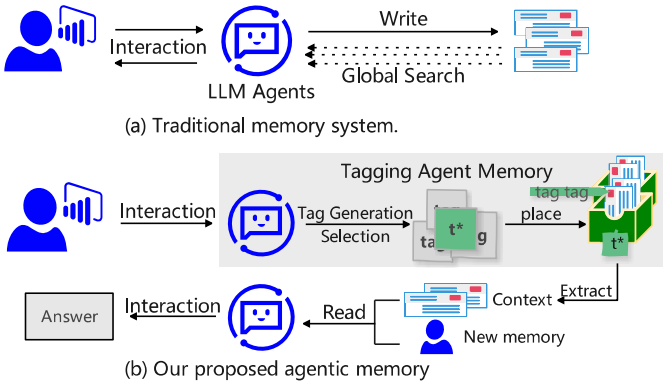

Fig. 1. This diagram contrasts traditional direct memory access (a) with our proposed system (b), where a dedicated Tagging Agent manages all memory interactions for the LLM.

In this paper, We argue that the key to scalable long-term memory lies in making explicit indexing decisions during the storage phase, rather than deferring the organizational effort solely to the retrieval stage. To this end, we propose agentic memory tagging, a training-free framework that promotes semantic tags to active storage keys. By routing memories into deterministic "tag boxes" during the storage phase, our method avoids exhaustive global searches and ensures structural stability through canonicalization and in-box consolidation. As shown in Figure 1, to ensure the stability and efficiency of the index, the agent incorporates two mechanisms for maintenance. First, a self-consolidating interface for tags aligns synonymous labels with shared canonical keys, which prevents fragmentation caused by variations in lexis. Second, when the number of entries within a tag box exceeds a predefined limit, redundant entries are merged through a process of consolidation to preserve both the density of critical information

*Correspondence: fanmiao@bigc.edu.cn

and its provenance. Queries can be efficiently directed to the most relevant tag boxes through inverted indices, allowing fast and coherent access to memory. This design ensures that the indexing structure remains stable even as the volume of stored memory increases. Our evaluation of the LOCOMO benchmark demonstrates that this module, which can be deployed without modifications to the underlying LLM, significantly reduces redundancy and improves coherence compared to retrieval-focused baselines.

Our contributions are summarized as follows: 1) We identify a critical bottleneck in long-term conversational memory: the deferral of semantic indexing to the retrieval phase and the resulting index fragmentation and storage redundancy; 2) We propose agentic memory tagging, a lightweight framework that enforces explicit semantic indexing during the storage phase. By employing canonicalization and box-level consolidation, it utilizes a primary tag as a deterministic storage key to proactively organize memory; 3) We demonstrate on the LOCOMO benchmark that our approach substantially reduces index redundancy and fragmentation, thereby improving retrieval coherence and efficiency while remaining inherently training-free and plug-and-play.

II. Related Work

Our work intersects research on long-context LLMs, external memory systems, and retrieval-centric organization. Substantial prior work focused on extending the context window through efficient KV-cache management and hierarchical architectures [1], [15], [16]. These techniques help process long inputs by managing the internal context, but do not solve the problem of organizing information across different sessions. This challenge has motivated the development of external memory systems such as Mem0, CDMem, and A-Mem, which explore scalable memory extraction, context-dependent indexing, and memory structures [13], [17], [18]. However, current methods for managing these memory stores focus heavily on retrieval to improve evidence structure, including adaptive RAG and retrieval techniques based on graphs or neural networks [3], [19], [20]. A common characteristic of these systems is that memory organization is largely deferred to the retrieval phase. Whether utilizing similarity search, link traversal, or graph-based querying, the system must process often weakly structured memories during the retrieval phase to recover coherent context.This requirement, combined with the accumulation of equivalent information in diverse surface forms, leads to index fragmentation and storage redundancy. In contrast, we fill this gap by establishing a stable indexing structure during the storage phase [7]. Our approach proactively organizes memories by their semantic essence in the storage phase, thereby shifting the primary responsibility from the retrieval phase to the storage phase.

III. Problem Formulation

We consider a persistent LLM agent that interacts with its environment over extended horizons and continuously accumulates experiences in an external memory store. Let $M = \{m_1, m_2, \dots, m_t, \dots\}$ denote the memory collection, where each m_i is a textual record derived from past interactions, observations, or reflections. The agent must retrieve relevant memories via a retrieval function $R(q, M)$ that returns contextual information for a given query q. In prevalent retrieval-centric paradigms, retrieval is implemented as a global similarity search over the entire memory space:

$$R_{global}(q, M) = \text{Top-}k_{m \in M}[\text{sim}(e(q), e(m))], \qquad (1)$$

where $e()$ denotes a text embedding function and $\text{sim}(,)$ is a similarity measure. Although effective in small scale settings, this formulation exhibits fundamental limitations when $|M|$ grows over long-term interaction. We identify two structural issues that arise from deferring semantic organization to the retrieval phase:

Consider a user who mentions the same preference in multiple sessions: “I like my cat”, “Siamese is noisy at night”. Retrieval-centric memory systems typically store these as separate entries associated with different surface forms (“cat”, “Siamese”) or store a single entry multiple times in different locations, leading to fragmented indexing and redundant storage. In contrast, our method canonicalizes these tags to a shared semantic key (e.g., pet-cat) during the storage phase, deterministically placing them into the same tag box. This enables a coherent aggregation of knowledge while avoiding fragmentation as memory grows: Specifically, we introduce a structured indexing space defined by a set of canonical semantic keys C. A storage indexing function:

$$I_w : M \to C \qquad (2)$$

where w_i are tunable weights. Assign each memory to a canonical key, inducing a partition over the memory:

$$B(c) = \{m \in M \mid I_w(m) = c\}, \quad M = \biguplus_{c \in C} B(c). \qquad (3)$$

This formulation allows us to reinterpret memory not as an unstructured flat collection, but as a dynamically evolving partitioned space. The quality and stability of this partition determine the long-term behavior of the memory system. To this end, we aim to construct a partition structure that satisfies the following criteria:1) Semantic Coherence. Memories that express the same underlying concept should be deterministically mapped to the same partition. 2) Structural Stability. The number and distribution of partitions should evolve smoothly as new memories arrive, avoiding uncontrolled growth of the index space. 3) Retrieval Efficiency. Retrieval complexity should depend primarily on the number of relevant partitions rather than on the total memory number, enabling sub-linear scaling with respect to $|M|$. This formulation establishes a principled optimization target: designing mechanisms for I_w and partition maintenance. The goal is to preserve a stable indexing structure that remains computationally efficient and semantically organized over long-term interaction.

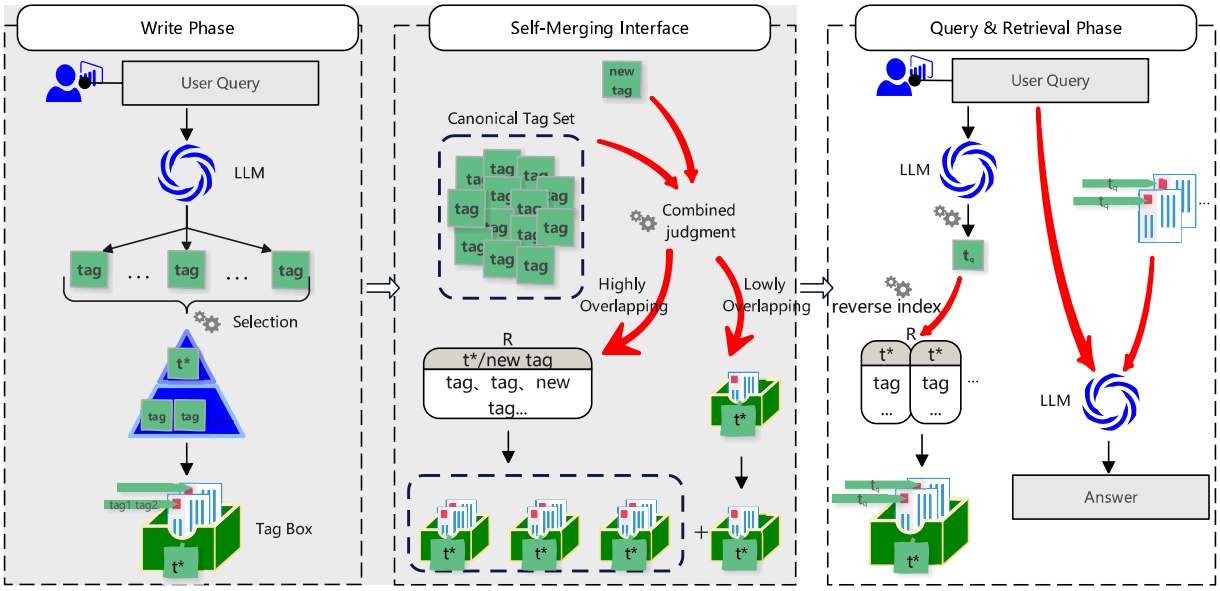

Fig. 2. This diagram illustrates the workflow of agentic memory tagging framework. The left side shows storage-phase processing (offline): semantic tagging, canonicalization, deterministic placement, and consolidation. The right side depicts retrieval-phase operations (online), where queries are routed to relevant tag boxes via the index, enabling efficient focused search instead of global scanning.

IV. Proposed Solution

We propose an agentic memory tagging framework that manages external memory by explicitly indexing semantics during the storage phase. As illustrated in Figure 2, the architecture consists of three components that form a closed-loop memory management pipeline: (1) semantic tagging with deterministic placement, (2) canonicalization, intra-box consolidation, and (3) indexed retrieval. The overall design follows a single guiding principle: impose semantic structure in the storage phase, so that long-term memory remains organized, stable, and efficient as it continuously grows.

A. Semantic Tagging and Deterministic Placement

For each incoming memory m_t, the agent generates a set of concise candidate tags for the noun phrase $T(m_t) = \{t_1, \ldots, t_k\}$ via constrained LLM prompting. The core mechanism is to select a single primary tag t^* as the explicit storage key through a scoring function:

$$t^* = \arg\max_{t \in T(m_t)} S(t \mid m_t), \tag{4}$$

where the scoring function integrates multiple objectives:

$$S(t \mid m) = \sum_{i=1}^{4} w_i \cdot \phi_i(t, m). \tag{5}$$

This formulation aims to obtain tags with appropriate semantic granularity, avoiding overly generic or overly specific keys. For instance, if user interactions are heavily focused on a single topic (e.g., "cats"), the box balancing penalty ($-log(1 + |B(c_t)|)$) in Eq. (6) progressively penalizes the generic tag as its box fills. This mechanism forces the LLM to "discover" and select more granular semantic levels (e.g., "cat health" or "cat diet") as the primary storage key. The feature components are defined as:

$$\phi_i(t, m) = \begin{cases} \text{sim}(e(m), e(t)), & \text{(semantic relevance)} \\ \log(1 + f(c_t)), & \text{(key stability)} \\ -\log(1 + |B(c_t)|), & \text{(box balancing)} \\ -\text{overlap}(t, T(m)), & \text{(tag diversity)} \end{cases} \tag{6}$$

where $c_t = \text{canonicalize}(t)$, $f(c_t)$ denotes the historical frequency of the canonical key.

Memory is deterministically placed according to the following:

$$I_w(m_t) = c_{t^*}, \quad m_t \in B(c_{t^*}). \tag{7}$$

This mechanism explicitly establishes semantic organization in the storage phase. By elevating the primary tag to a first-class structural key, memories that share core semantics are grouped into the same tag box by construction. In contrast to conventional systems where tags merely as auxiliary annotations, our design turns tags into the cornerstone of the indexing structure. Moreover, the multi-objective formulation in Eq. (5) provides controllability over the long-term evolution of the system. Specifically, the stability term favors the reuse of existing keys to avoid index explosion, while the balancing term prevents any single box from becoming overly crowded. This combination allows the system to maintain a stable, organized structure even as new interactions occur. Secondary tags serve as auxiliary metadata to enable fine-grained filtering within the box, ensuring the selection of highly relevant memories after routing via the primary key and index R.

B. Canonicalization: Alias-to-Canonical Mapping

To prevent index fragmentation caused by lexical variation, the agent maintains a dynamic canonicalization map A : SurfaceTag $\rightarrow C$: where a "surface tag" is defined as the raw, unnormalized lexical phrase extracted directly from the memory text:

$$A(t) = \begin{cases} c^{\dagger}, & \text{if } \max_{c \in C} \text{sim}(e(t), e(c)) \geq \tau, \\ c_{\text{new}}, & \text{otherwise,} \end{cases} \tag{8}$$

where $c^{\dagger} = \arg\max_{c \in C} \text{sim}(e(t), e(c))$ and τ is a similarity threshold. Values will be detailed later.

Canonicalization acts as a long-term regularization mechanism in the index space. Without this alignment, semantically equivalent tags such as "pet cat", "kitten", and "my cat" would gradually introduce separate keys, causing the index to fragment. By mapping new surface forms to existing canonical keys, the agent ensures that the index reflects conceptual differences, rather than superficial linguistic variations. Over time, this process induces a stable and compact vocabulary of canonical keys, allowing the memory structure to evolve smoothly instead of expanding uncontrollably.

C. Intra-Box Consolidation

When the size of a box $|B(c)|$ exceeds a threshold Γ, a consolidation process is triggered. By prompting the LLM, the agent synthesizes the contents within the box into a compact knowledge entry K_c, specifically instructed to perform semantic deduplication, resolve temporal conflicts (e.g., by prioritizing more recent or consistent statements) , and abstract stable factual knowledge. In concrete terms, consolidation operates on the set of memories and their associated auxiliary tags in $B(c)$. Initially, the agent clusters entries by their embedding similarity and overlapping tags, flagging these groups as potential redundancies. Next, the LLM processes these clusters to generate a unified representation. Retention of unique details while filtering out repetitive or semantically equivalent content. In this way, redundancy is resolved through semantic synthesis. The resulting compact entry K_c is stored as a memory within the box, and the original memories are not discarded. The system explicitly links K_c to the original memories for provenance:

$$\text{Prov}(K_c) = \{m_1, m_2, \ldots, m_{|B(c)|}\}, \tag{9}$$

which are stored as metadata within the box. This design ensures traceability: the agent can always inspect original sources when needed, preserving transparency and supporting explainability. Consolidation also influences retrieval behavior. In the retrieval phase, K_c serves as the primary representative of the box, allowing fast access to the knowledge about high density. If granular details are required, the agent can further expand the retrieval to the original linked memories through the provenance structure. This hierarchical organization allows the system to support both efficient high-level recall and detailed inspection, while preventing the accumulation of unbounded low-value redundancy over long-term interaction.

D. Indexed Retrieval Process

Retrieval utilizes the index constructed during the storage phase and proceeds in two steps: 1) Query Canonicalization. A primary tag t_q is extracted from the query q and then assigned to a canonical key $c_q = A(t_q)$. 2) Focused Search. Retrieval is restricted to relevant boxes using the reverse index:

$$R(t) \rightarrow \{c \mid \exists m \in B(c) \text{ with tag } t\}. \tag{10}$$

Items within the selected boxes are scored by:

$$\text{score}(m, q) = \lambda \cdot \text{sim}(e(m), e(q)) + (1 - \lambda) \cdot \exp(-\Delta t(m)/\tau_r), \tag{11}$$

where $\Delta t(m)$ denotes the temporal interval between the memory and the current query. The top-k items are returned as contextual support. Compared with global retrieval strategies that require scanning the entire memory store, this indexed retrieval mechanism substantially reduces the search space. The expected retrieval cost depends mainly on the size of the relevant boxes, rather than on $|M|$. Canonicalization and consolidation constrain the index's long-term growth, thereby ensuring stable retrieval efficiency despite memory expansion.

V. Experiments

We perform experiments to evaluate whether the proposed agentic memory tagging reaches its intended design goals: constructing a stable indexing structure, reducing redundancy and fragmentation in long-term memory, and improving retrieval coherence and efficiency. All experiments are conducted on the LOCOMO benchmark, which is specifically designed to evaluate long-range memory behavior in conversational agents.

A. Experimental Setup

We evaluate on the LOCOMO benchmark, which contains long-span multi-session dialogs with an average context length of 9K tokens and 7,512 pairs of questions and answers. The questions cover diverse reasoning types, This benchmark evaluates whether a memory system remains coherent and structured as the interaction length increases. We compare our method with representative systems, including Mem0, A-Mem, and CDMem. These baselines cover both retrieval-centric and structured memory paradigms, enabling a comprehensive evaluation of the storage phase's semantic organization.

To evaluate answer quality supported by context retrieved, we adopt standard metrics used in prior LOCOMO work, including F1 for factual overlap and BLEU-1 for lexical consistency [21]. In addition to response quality, we directly evaluate the structural properties emphasized in this work by analyzing system-level behaviors, including index growth dynamics, storage redundancy, and retrieval latency as memory size increases. These measurements directly correspond to the claims made in the abstract and the problem formulation. We use all-mpnet-base-v2 for embedding computation. The canonicalization threshold $\tau = 0.85$ is empirically determined to balance semantic alignment and over-merging within general conversational contexts. However, for highly technical or

TABLE I
MAIN QUANTITATIVE RESULTS ON THE LOCOMO BENCHMARK.

Method	Single-hop	Multi-hop	Temporal	Open-domain	Adversarial	Overall
Mem0 [17]	64.1	50.2	47.0	59.1	51.8	43.7
A-Mem [13]	65.4	52.6	49.8	60.3	53.1	44.9
CDMem [18]	66.0	53.4	50.5	61.0	54.2	45.2
Ours	**69.3**	**58.4**	**55.1**	**63.8**	**56.7**	**48.7**

Note: Single-hop, Multi-hop, Temporal, Open-domain, and Adversarial tasks are evaluated by the F1 score; the Overall performance is evaluated by the BLEU-1 score.

domain-specific knowledge bases where synonymy is less flexible, this threshold necessitates upward adjustment to prevent the over-generalization of distinct concepts. The consolidation trigger $\Gamma = 50$ limits redundancy growth without premature summarization. The weights in Eq. (5) are set to (0.4, 0.3, 0.2, 0.1), prioritizing semantic relevance and key stability. All LLM-based components are implemented using GPT-4 under uniform conditions to ensure fairness.

B. Main Results

Table I reports the main quantitative results on LOCOMO in different categories of questions. The proposed method consistently achieves strong performance across all categories and shows clear advantages on tasks that are most sensitive to long-term coherence, such as multi-hop and temporal reasoning. Compared with A-Mem, our method achieves a substantial improvement across these categories, reflecting that the retrieved context is more semantically coherent and structurally organized. These findings support the central thesis of this research.

Beyond answer quality, we analyze how the memory structure evolves as the interaction progresses. Figure 3 reports the number of index keys as a function of the accumulation process. Baseline systems exhibit nearly linear growth in the number of indexed entries, reflecting their tendency to continuously generate new but weakly structured records. In contrast, our method shows clear stabilization after an initial growth stage. This behavior is achieved by aligning semantically similar memories into existing tag boxes through canonicalization and consolidation, confirming that the proposed agent maintains a stable indexing structure as memory scales.

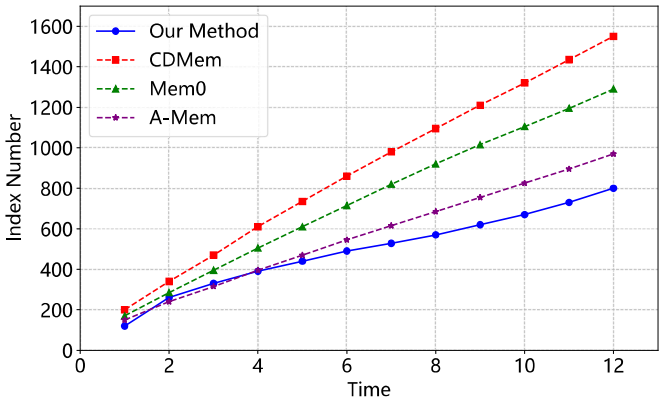

Fig. 3. Index number growth over time. Our method shows early growth followed by stabilization, while baselines exhibit continuous growth due to fragmentation.

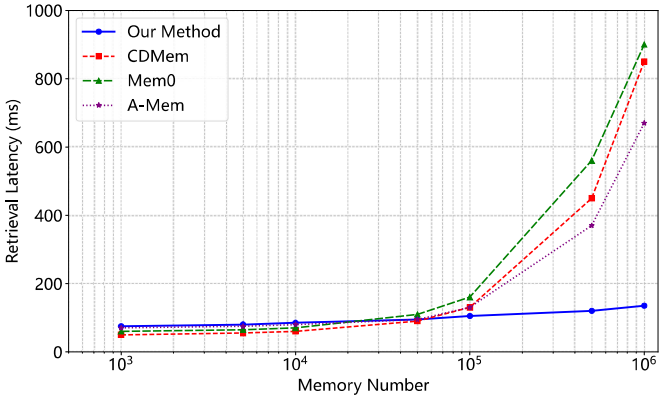

Fig. 4. Retrieval latency vs. memory size. Our approach maintains low and stable latency as memory grows, while baselines show clear increases.

C. Scalability and Efficiency

We further evaluate scalability by measuring the retrieval latency as the memory volume increases. Figure 4 shows the average retrieval time as the accumulated memory grows from 1K to 1M entries. Baseline systems exhibit increasing latency as memory grows, reflecting the cost of a weakly structured or global search. In contrast, our method maintains consistently low retrieval latency with only mild growth. This behavior arises because queries are routed to a small number of relevant tag boxes via the storage phase index, rather than requiring a search over the entire memory space. This result provides empirical validation for the theoretical motivation introduced in Section III. Because canonicalization constrains index expansion and consolidation limits redundancy within each box, the effective search space remains bounded even as total memory grows. The combination of these mechanisms enables the system to preserve both retrieval efficiency and structural stability under long-term accumulation.

D. Component Analysis

Table II reports ablation results that evaluate the contribution of individual components. Removing canonicalization leads to degradation in retrieval performance and a substantial increase in index number, indicating that alias-to-canonical alignment is essential for preventing fragmentation. Disabling consolidation significantly increases storage volume, confirm-

TABLE II
ABLATION STUDY ON LOCOMO. WE REPORT RETRIEVAL QUALITY AND STRUCTURAL PROPERTIES OF THE MEMORY.

Configuration	F1	Index number	Storage number	Redundancy (%)
w/o Canon.	53.2	4,980	6,120	42.5
w/o Cons.	57.6	2,140	5,890	38.1
w/o Det. Tag	50.8	3,760	4,970	35.4
Ours	**58.9**	**1,420**	**2,310**	**17.6**

ing that redundancy naturally accumulates even when semantic grouping is correct and that consolidation plays a critical role in maintaining compact memory. The replacement of deterministic tag selection with random selection leads to consistent performance degradation, demonstrating that the semantic placement of the storage-phase is the foundation of the proposed framework.

Taken together, these results show that semantic tagging with deterministic placement, canonicalization, and intra-box consolidation work jointly to produce a memory system that is more structured, more compact, and more efficient than retrieval-centric alternatives. Although the storage phase introduces slightly higher overhead due to explicit indexing, this cost is amortized over time and results in substantially faster and more stable retrieval, with tagging introducing a small constant overhead per memory and consolidation triggered sparsely when $|B(c)|$ exceeds Γ. The experimental evidence consistently supports the central thesis of this work: the agentic memory tagging framework, which shifts semantic organization from the retrieval phase to the storage phase, leads to more robust long-term memory performance.

VI. CONCLUSION

This work presents agentic memory tagging, a framework that improves long-term memory for LLM agents by organizing memories during the storage phase rather than deferring structure to retrieval. The agent constructs a stable memory index through semantic tagging, canonicalization, and intra-box consolidation, reducing redundancy and maintaining coherence as memory grows. Experiments on the LOCOMO benchmark demonstrate improved retrieval quality. Although shifting computation to the storage phase introduces additional ingestion overhead, the framework provides an efficient and structured solution for managing long-term interactions. Although this amortized cost is acceptable in cloud-based deployments, future implementations in resource-constrained edge environments will require a deeper theoretical examination of the trade-offs between storage-phase computational overhead and real-time interaction limits. Future work may explore richer tag hierarchies and adaptive thresholding for greater flexibility.

ACKNOWLEDGMENTS

This work was supported by the Fundamental Research Funds for the Municipal Universities of Beijing (No. 04190125001/015).

REFERENCES

[1] Møller, A.G., et al.: The parrot dilemma: Human-labeled vs. llm-augmented data in classification tasks. In: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 179–192 (2024)
[2] Alizadeh, K., et al.: Llm in a flash: Efficient large language model inference with limited memory. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 12562–12584 (2024)
[3] Yuan, R., et al.: Personalized large language model assistant with evolving conditional memory. In: Proceedings of the 31st International Conference on Computational Linguistics. pp. 3764–3777 (2025)
[4] Jia, Z., et al.: Evaluating the long-term memory of large language models. In: Findings of the Association for Computational Linguistics: ACL 2025. pp. 19759–19777 (2025)
[5] Chen, H.M., et al.: Hardware-aware parallel prompt decoding for memory-efficient acceleration of llm inference. arXiv preprint arXiv:2405.18628 (2024)
[6] Wang, B., et al.: Unveiling privacy risks in llm agent memory. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 25241–25260 (2025)
[7] Jin, M., et al.: Disentangling memory and reasoning ability in large language models. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1681–1701 (2025)
[8] Jeong, S., et al.: Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. arXiv preprint arXiv:2403.14403 (2024)
[9] Zhang, C., et al.: Working memory identifies reasoning limits in language models. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 16896–16922 (2024)
[10] Barry, M., et al.: Graphrag: Leveraging graph-based efficiency to minimize hallucinations in llm-driven rag for finance data. In: Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK). pp. 54–65 (2025)
[11] Ding, J., et al.: Msg-llm: A multi-scale interactive framework for graph-enhanced large language models. In: Proceedings of the 31st International Conference on Computational Linguistics. pp. 9687–9700 (2025)
[12] Fang, J., et al.: Trace the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation. arXiv preprint arXiv:2406.11460 (2024)
[13] Xu, W., et al.: A-mem: Agentic memory for llm agents. arXiv preprint arXiv:2502.12110 (2025)
[14] Xu, M., et al.: Memory-augmented query reconstruction for llm-based knowledge graph reasoning. arXiv preprint arXiv:2503.05193 (2025)
[15] He, Z., et al.: Hmt: Hierarchical memory transformer for efficient long context language processing. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). pp. 8068–8089 (2025)
[16] Kim, M., et al.: Infinipot: Infinite context processing on memory-constrained llms. arXiv preprint arXiv:2410.01518 (2024)
[17] Chhikara, P., et al.: Mem0: Building production-ready ai agents with scalable long-term memory. arXiv preprint arXiv:2504.19413 (2025)
[18] Gao, P., et al.: An efficient context-dependent memory framework for llm-centric agents. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track). pp. 1055–1069 (2025)
[19] Venkatraman, S., et al.: Collabstory: Multi-llm collaborative story generation and authorship analysis. In: Findings of the Association for Computational Linguistics: NAACL 2025. pp. 3665–3679 (2025)
[20] Zhang, K., et al.: Llm-based medical assistant personalization with short-and long-term memory coordination. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). pp. 2386–2398 (2024)
[21] Maharana, A., et al.: Evaluating very long-term conversational memory of LLM agents. In: Ku, L.W., Martins, A., Srikumar, V. (eds.) Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 13851–13870. Association for Computational Linguistics, Bangkok, Thailand (Aug 2024).