Lightweight LLM Agent Memory with Small Language Models

Jiaquan Zhang[1], Chaoning Zhang[1,*], Shuxu Chen[2], Zhenzhen Huang[1], Pengcheng Zheng[1], Zhicheng Wang[1], Ping Guo[3], Fan Mo[4], Sung-Ho Bae[2], Jie Zou[1], Jiwei Wei[1], Yang Yang[1]
[1]University of Electronic Science and Technology of China;
[2]Kyung Hee University; [3]City University of Hong Kong; [4]University of Oxford
***Corresponding author:** chaoningzhang1990@gmail.com

Abstract

Although LLM agents can leverage tools for complex tasks, they still need memory to maintain cross-turn consistency and accumulate reusable information in long-horizon interactions. However, retrieval-based external memory systems incur low online overhead but suffer from unstable accuracy due to limited query construction and candidate filtering. In contrast, many systems use repeated large-model calls for online memory operations, improving accuracy but accumulating latency over long interactions. We propose *LightMem*, a lightweight memory system for better agent memory driven by Small Language Models (SLMs). *LightMem* modularizes memory retrieval, writing, and long-term consolidation, and separates online processing from offline consolidation to enable efficient memory invocation under bounded compute. We organize memory into short-term memory (STM) for immediate conversational context, mid-term memory (MTM) for reusable interaction summaries, and long-term memory (LTM) for consolidated knowledge, and uses user identifiers to support independent retrieval and incremental maintenance in multi-user settings. Online, *LightMem* operates under a fixed retrieval budget and selects memories via a two-stage procedure: vector-based coarse retrieval followed by semantic consistency re-ranking. Offline, it abstracts reusable interaction evidence and incrementally integrates it into LTM. Experiments show gains across model scales, with an average F1 improvement of about 2.5 on LoCoMo, more effective and low median latency (83 ms retrieval; 581 ms end-to-end).

1 Introduction

LLM-driven agents excel at long-term dialogue, multi-step reasoning, and task-oriented interaction (Zhang et al., 2026c; Huang et al., 2024; Zhu et al., 2025; Zhang et al., 2026a,b, 2025a). To maintain cross-turn consistency beyond the context window, many systems augment agents with external memory (Lee et al., 2024; Xu et al., 2025; Hu et al., 2025; Wang et al., 2026). Long-term memory supports continual learning and planning.

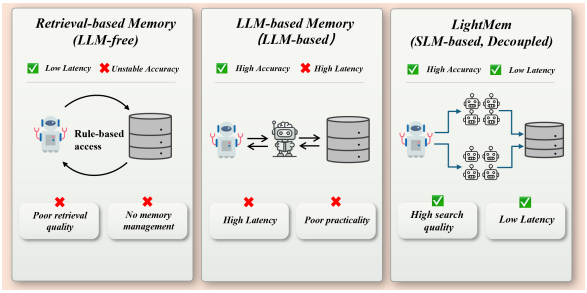


Figure 1: *LightMem* combines enhanced retrieval with SLMs, achieving high retrieval accuracy while significantly reducing online latency compared to retrieval-based and LLM-based memory systems.

Existing memory systems can be broadly divided into two categories. One line is based on retrieval–driven external memory, where past interactions are compressed into retrievable entries (Asai et al.; Maharana et al., 2024) and the top-K memories are recalled via similarity search (Zhong et al., 2024). These methods are efficient, but limited query construction and candidate filtering often introduce retrieval noise, resulting in unstable answer accuracy. Another line of methods (Tan et al., 2025; Zhang et al., 2025b) introduces LLM-driven memory operations on top of external memory, repeatedly invoking large models for memory writing, retrieval and controlling. As these operations are typically implemented via repeated model invocations, they can introduce non-trivial runtime overhead over long interactions (Packer et al., 2023; Park et al., 2023). This contrast suggests a natural separation: keep high-frequency online memory decisions lightweight and controllable, while deferring heavy abstraction and consolidation to offline processing (see Figure 1). Recent advances in SLMs (Magister et al., 2023; Han et al., 2024)

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12914–12929
July 2-7, 2026

make this separation practical. SLMs can reliably handle high-frequency, structured decision tasks in online memory processing, enabling different memory stages to be assigned to models of appropriate scale (Liu et al., 2025). Online memory handling typically consists of structured subtasks (such as intent routing (Zhang et al., 2024), query construction (Hong and He, 2025) and semantic filtering (Hatalis et al., 2023)). These tasks place greater emphasis on predictable behavior and low overhead than on maximal generative capacity. In this setting, lightweight SLMs provide a suitable choice (Sinha et al., 2025) for online control and filtering, while heavier abstraction and consolidation can be deferred to offline processing. Overall, existing memory systems face an efficiency–effectiveness trade-off. Moreover, SLMs are not a silver bullet due to limited capacity and representation. Bridging this gap requires strengthening the external memory pipeline, especially how memories are written and retrieved.

To address the above questions, we propose *LightMem*, a lightweight memory system for LLM agents that models long-term memory as an incrementally evolvable process via specialized SLMs. *LightMem* modularizes and decouples query parsing, memory retrieval, memory writing, and long-term consolidation, enabling independent optimization and elastic scaling of each component. This separation allows for lightweight online processing to be distinct from offline consolidation, achieving efficient memory use under tight compute budgets and supporting long-term evolution. Memories are organized into STM, MTM, and LTM stores based on temporal and access characteristics. User-identity metadata is embedded in each memory unit to enforce user-level logical isolation, balancing privacy, consistency, and scalability. We propose a modular online–offline memory pipeline in which distinct SLMs specialize in complementary memory operations. Online, LightMem uses three specialized small language model modules: a Controller (SLM-1) for intent and query planning, a Selector (SLM-2) for candidate verification and compression, and a Writer (SLM-3) for incremental memory writing. Specifically, the Controller rewrites the user input into intent-conditioned hypothetical queries (HQs) and allocates a fixed Top-K budget across MTM and LTM. Given the retrieved candidates, the Selector performs metadata-constrained prefiltering followed by semantic-consistency based re-ranking to output the final Top-K memories. After each turn, the Writer summarizes the interaction into compact MTM entries and maintains MTM incrementally. Long-term abstraction and consolidation are handled offline by a large-context model, keeping the online path lightweight. Our main contributions are summarized as:

- We propose *LightMem*, a lightweight memory system collaboratively driven by SLMs. Different SLMs handle lightweight, high-frequency online memory operations (query construction, retrieval, and writing), while heavier abstraction and consolidation are deferred to offline processing.
- We propose a two-stage memory querying design. Given a user query, *LightMem* first narrows down the candidate set with fast retrieval, and then applies semantic-level verification to select the truly relevant memories.
- We evaluate *LightMem* on LoCoMo and DialSim, demonstrating gains across model scales, including an average F1 improvement of about 2.5 on LoCoMo, improved semantic consistency on DialSim and low median latency (83 ms retrieval; 581 ms end-to-end).

2 Related Work

2.1 LLM Memory Systems

Prior work can be broadly categorized into retrieval-based memory and LLM-driven memory operations. **Retrieval-based Memory.** MemoryBank (Zhong et al., 2024) supports personalized long-term dialogue by storing summarized user events in an external memory and retrieving relevant entries for each query, with forgetting to control growth. MemGPT (Packer et al., 2023) treats the context window as virtual memory and performs paging between the prompt and an external store, with runtime eviction and on-demand retrieval. ReadAgent (Lee et al., 2024) similarly relies on retrieval over an external cache: it indexes compressed gists and performs on-demand lookup to fetch supporting evidence when needed. **LLM-driven Memory.** HiAgent (Hu et al., 2025) manages in-trial working memory by chunking trajectories into subgoals and summarizing past steps with LLMs. A-MEM (Xu et al., 2025) further builds self-organizing memory networks through LLM-driven note-taking and automatic linking, but typically does not emphasize strict online/offline decoupling under resource

constraints. These lines expose a recurring trade-off between lightweight but noisy retrieval-based memory and more effective yet costly LLM-driven online memory operations. *LightMem* addresses it with SLM-based lightweight online control and filtering, coupled with improved external memory writing and retrieval under a fixed budget.

3 Method

As shown in Figure 2, *LightMem* uses specialized SLMs to modularize memory operations, separating lightweight online querying from offline long-term consolidation.

3.1 Problem Setup and Preliminaries

We consider a multi-turn dialogue setting with multiple users. At turn t, a user provides an input x_t, and the model generates a response y_t. Due to the limited context capacity of SLMs, the accessible dialogue context at turn t is denoted by C_t, typically a truncated window of recent turns. *LightMem* maintains a user-scoped memory store M_u with user-level logical isolation via user identifiers (see §3.2 for the definition of STM/MTM/LTM). Given (x_t, C_t), the system retrieves a memory set $R_t \subseteq M_u$ with $|R_t| \leq K$ and uses R_t to assist in generating y_t. The specific symbols are provided in the Appendix 8.1.

3.2 Memory Stores

STM. STM is implemented as the SLM context window that buffers the most recent interaction sequence in the current session. It serves only as working memory and is updated turn by turn within the prompt; STM is neither persisted nor retrieved.

MTM. MTM is the sole carrier of personalized episodic memory. It stores a set of memory items, each consisting of (i) a concise semantic summary, (ii) temporal information and access statistics, (iii) an embedding for similarity-based retrieval, and (iv) a user identifier for strict per-user isolation.

LTM. LTM stores de-identified, user-agnostic semantic knowledge distilled offline from high-value MTM episodes. It contains no raw personal episodes or user identifiers; instead, it represents stable facts, domain regularities, and cross-user trends. LTM is organized as a lightweight graph-structured knowledge base to support multi-hop reasoning and knowledge sharing across tasks.

3.3 Intent Modeling and Retrieval Control

Given (x_t, C_t), SLM-1 serves as a lightweight retrieval controller rather than a retriever. It converts the raw input into a structured retrieval plan that specifies: (i) what to search (queries), (ii) how to constrain the search (metadata filters), and (iii) how many items to return under a fixed Top-K budget. Concretely, SLM-1 first infers coarse intent attributes, e.g., whether the query depends more on recent episodic details or long-term stable knowledge, and whether strong personalization is needed. These attributes are used only for retrieval planning (query rewriting, routing, and budgeting), not for answer generation. SLM-1 then rewrites the input into a set of HQs $\{q_t^{(i)}\}$ and outputs metadata constraints ϕ_t (user identifier, optional time window, and type tags) to reduce noise and enforce user-level isolation. Finally, it issues the retrieval request:

$$\mathcal{Q}_t = \left\langle \{q_t^{(i)}\},\ \phi_t,\ K \right\rangle. \tag{1}$$

This request is consumed by §3.4.

3.4 Two-Stage Retrieval

Given $\mathcal{Q}_t$, SLM-2 executes a two-stage retrieval procedure and returns a memory set R_t that satisfies the Top-K constraint:

$$R_t = \textsc{Retrieve}(\mathcal{Q}_t), \qquad |R_t| \leq K. \tag{2}$$

The retrieved memories R_t are appended to the context to condition response generation for y_t.

For a single user query, SLM-1 may generate n HQs, denoted by $\{q_t^{(i)}\}_{i=1}^{n}$, to improve recall coverage. We impose a single final return budget K and adopt a double-budget rule in the coarse stage: the total number of candidates returned by Stage 1 is fixed to $2K$.

Stage 1: metadata-constrained coarse retrieval. Under metadata constraints ϕ_t (including the user identifier, an optional time window, and type/source tags), the system performs vector-based coarse retrieval for each HQ. The Stage 1 candidate budget is evenly split across HQs:

$$\sum_{i=1}^{n} K_1^{(i)} = 2K, \qquad K_1^{(i)} = \frac{2K}{n}. \tag{3}$$

Let $C^{(i)}$ denote the candidate list returned for the i-th HQ. The aggregated candidate set is

$$C = \bigcup_{i=1}^{n} C^{(i)}, \qquad |C| = 2K. \tag{4}$$

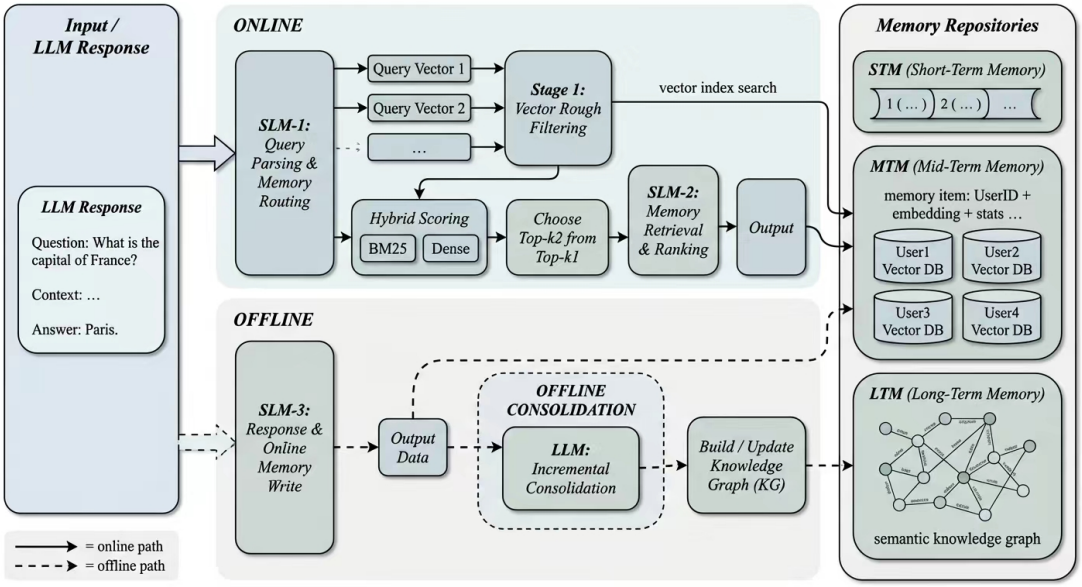


Figure 2: Multiple SLMs coordinate an online pathway for query-time routing and retrieval over STM/MTM, and an offline pathway that incrementally consolidates MTM into a graph-structured LTM.

Stage 1 is designed for coverage, acting as an efficient filter that compresses the large search space into a fixed-size candidate set.

Stage 2: semantic filtering and compression. In Stage 2, SLM-2 takes the HQ set $\{q_t^{(i)}\}_{i=1}^{n}$ and the Stage 1 candidates C (memory summaries with necessary structured metadata) and performs semantic consistency checking and relevance judgment. Instead of generating answers, SLM-2 conducts controlled selection on a fixed-size candidate pool: it keeps the most relevant half of $|C| = 2K$ candidates and outputs the final retrieval results

$$R_t \subseteq C, \qquad |R_t| \leq K \tag{5}$$

This two-to-one compression yields (i) stable computation with a fixed candidate size, (ii) semantic refinement beyond vector similarity by leveraging intent and metadata, and (iii) noise suppression by explicitly discarding roughly half of the candidates. The resulting Top-K memories R_t are provided to the downstream agent or response generator.

3.5 Memory Writing and Update

After generating y_t, SLM-3 extracts reusable user-relevant information from the current interaction, compresses it into a concise memory item, and appends it to MTM. To prevent redundancy and noise accumulation, highly repetitive items are merged or rewritten, while conflicting information is handled using temporal cues and evidence strength. To ensure consistently low-latency personalized retrieval, we enforce a capacity bound on MTM:

$$|M_u^{\mathrm{MTM}}| \leq B \tag{6}$$

When the bound is reached, MTM is maintained by evicting stale, low-utility items and further compressing redundant content.

3.6 Offline Consolidation

To avoid increasing online retrieval and writing latency, *LightMem* performs offline consolidation to distill high-value episodic evidence in MTM into de-identified, long-term semantic knowledge, enabling sustained evolution of LTM. An LLM handles this offline path with a large context window and is strictly decoupled from online operation. In each cycle, the LLM processes only an incremental batch (newly written or retrieval-reactivated MTM items and low-utility candidates flagged under MTM capacity pressure), rather than rebuilding MTM/LTM from scratch. It abstracts episodes into privacy-preserving knowledge candidates, performs similarity search over LTM to locate semantically nearest anchors, and incrementally inserts and links candidates within the local neighborhood, thereby maintaining LTM as lightweight graph-structured knowledge. Evidence accumulated over time further drives merge/update/drop decisions, while confidence decay is applied to weakly supported candidates to enable natural forgetting and to limit the impact of stale or incidental information on subsequent retrieval and reasoning. We provide

each component with algorithmic details deferred to Appendix 8.1.

4 Experiment

4.1 Experimental Setup

4.1.1 Datasets

To evaluate the effectiveness of *LightMem*, we conduct experiments on two datasets. LoCoMo (Maharana et al., 2024) is a benchmark designed to evaluate logical reasoning over extended conversational contexts. It contains long dialogues (on average 9K tokens) and covers five core task categories: Single-hop, Multi-hop, Temporal, Open-domain, and Adversarial. DialSim (Kim et al., 2024) is a dialogue simulation dataset derived from TV shows, containing 1,300 multi-party sessions spanning years.

4.1.2 Baselines

Baselines We select baselines that are representative in prior work on long-term memory, covering common designs for storing, retrieving, and updating dialogue histories and that can be evaluated under the same backbone and retrieval budget for a fair comparison: LoCoMo (Maharana et al., 2024), ReadAgent (Lee et al., 2024), MemoryBank (Zhong et al., 2024), MemGPT (Packer et al., 2023), and A-MEM (Xu et al., 2025). When selecting backbone models, a diverse array of LLMs is used to assess their generalization capabilities. These models encompass GPT-4o, GPT-4o-mini (Hurst et al., 2024), Qwen2.5 (available in 1.5B and 3B) (Bai et al., 2025), and Llama 3.2 (offered in 1B and 3B) (Dubey et al., 2024).

4.1.3 Metric

We evaluate response quality using both lexical-overlap and semantic metrics. Specifically, we report F1 and BLEU-1 to measure answer correctness and token-level overlap with reference responses, which are commonly used for short factual and span-like outputs. For DialSim (Kim et al., 2024), where multiple surface forms can express the same meaning, we additionally report ROUGE-L and METEOR to capture sequence-level overlap and soft token matching, and use SBERT similarity to quantify semantic consistency between generated and reference responses beyond n-gram matches. These metrics provide a reliable assessment of both precision and meaning retention in long-horizon dialogue scenarios.

4.1.4 Implement Details

To isolate the effect of memory, we use GPT-4o-mini as the fixed response generator for *LightMem* and all baselines. *LightMem*'s control plane uses locally deployed, quantized SLMs: SLM-1 for HQ generation and retrieval control, SLM-2 for semantic re-ranking and compression, and SLM-3 for online writing and MTM maintenance. For HQ generation, SLM-1 follows a structured prompt for query decomposition and routing; Table 1 summarizes the prompt logic and provides an example. We deploy quantized Llama-3.2-1B-Instruct via Ollama by default and additionally evaluate Qwen2.5-1.5B-Instruct for robustness. Offline consolidation is handled by a large-context LLM and is decoupled from the online path. We encode all queries and memory entries using all-MiniLM-L6-v2 (384 dimensions) and retrieve the top-10 nearest neighbors in the coarse vector retrieval stage. SLM-2 is fine-tuned with LoRA on 2,000 constructed (Query, Subgraph, Path) samples. We cap MTM capacity at $B = 10^4$ and apply dynamic pruning beyond this limit by evicting stale, low-utility items and merging/compressing redundant content. *LightMem* inference and latency measurements are conducted on a single NVIDIA RTX 4090 (24GB) GPU. Other unspecified settings follow A-MEM (Xu et al., 2025).

Table 1: Structured HQ prompt used by SLM-1 for query decomposition and routing.

HQ prompt step	**Example**
Detect missing info	Identify underspecified references (e.g., pronouns such as "it/that/he/the project"), implicit context dependencies, and vague time cues (e.g., "recently/last time/before").
Generate HQs	Rewrite the user request into one or more standalone HQs, optionally splitting intent into user-specific vs. factual queries, and making implied context (time/location) explicit.
Route and budget	Assign each HQ to MTM (user-specific) or LTM (general/public knowledge) and allocate a retrieval budget under a fixed top-K constraint.
Example	Input: "Recommend a dinner spot." The prompt yields two HQs: one asks about the user's preferred cuisines or dietary constraints (routed to MTM), and the other asks about highly rated nearby restaurants (routed to LTM).

Table 2: Main results on LoCoMo across question categories. We report F1 and BLEU-1 for each category, along with the token length of the effective context. Best results within each model block are in bold.

Model	Method	Single-hop		Multi-hop		Temporal		Open-domain		Adversarial		Token Length
		F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	
GPT-4o-mini	LoCoMo	40.36	29.05	25.02	19.75	18.41	14.77	12.04	11.16	**69.23**	**68.75**	16,910
	ReadAgent	9.67	7.66	9.15	6.48	12.60	8.87	5.31	5.12	9.81	9.02	643
	MemoryBank	6.61	5.16	5.00	4.77	9.68	6.99	5.56	5.94	7.36	6.48	432
	MemGPT	41.04	34.34	26.65	17.72	25.52	19.44	9.15	7.44	43.29	42.73	16,977
	A-MEM	44.65	37.06	27.02	20.09	45.85	36.67	12.14	12.00	50.03	49.47	2,520
	LightMem	**45.80**	**38.20**	**28.85**	**21.40**	**46.20**	**37.10**	**13.50**	**12.90**	54.50	52.10	1,150
GPT-4o	LoCoMo	**61.56**	**54.19**	28.00	18.47	9.09	5.78	16.47	14.80	**52.61**	**51.13**	16,910
	ReadAgent	12.46	10.29	14.61	9.95	4.16	3.19	8.84	8.37	6.81	6.13	805
	MemoryBank	8.28	7.10	6.49	4.69	2.47	2.43	6.43	5.30	4.42	3.67	569
	MemGPT	60.16	53.35	30.36	22.83	17.29	13.18	12.24	11.87	34.96	34.25	16,987
	A-MEM	48.43	42.97	32.86	23.76	39.41	31.23	17.10	15.84	36.35	35.53	1,216
	LightMem	49.80	44.20	**34.50**	**25.10**	**40.10**	**32.00**	**18.40**	**16.90**	41.50	39.80	680
Qwen2.5-1.5B	LoCoMo	11.15	8.67	9.05	6.55	4.25	4.04	9.91	8.50	40.38	40.23	16,910
	ReadAgent	10.13	7.54	6.61	4.93	2.55	2.51	5.31	12.24	5.42	27.32	752
	MemoryBank	13.42	11.01	11.14	8.25	4.46	2.87	8.05	6.21	36.76	34.00	284
	MemGPT	9.56	7.34	10.44	7.61	4.21	3.89	13.42	11.64	31.51	28.90	16,953
	A-MEM	23.63	19.23	18.23	11.94	24.32	19.74	16.48	14.31	46.00	43.26	1,300
	LightMem	**25.10**	**20.50**	**21.50**	**13.80**	**25.60**	**20.80**	**17.20**	**15.10**	**49.80**	**46.50**	720
Qwen2.5-3B	LoCoMo	7.03	5.69	4.61	4.29	3.11	2.71	4.55	5.97	16.95	14.81	16,910
	ReadAgent	3.25	2.51	2.47	1.78	3.01	3.01	5.57	5.22	15.78	14.01	776
	MemoryBank	4.11	3.32	3.60	3.39	1.72	1.97	6.63	6.58	13.07	10.30	298
	MemGPT	7.26	5.52	5.07	4.31	2.94	2.95	7.04	7.10	14.47	12.39	16,961
	A-MEM	17.23	13.12	12.57	9.01	27.59	25.07	7.12	7.28	27.91	25.15	1,137
	LightMem	**19.80**	**15.20**	**15.80**	**11.20**	**29.10**	**26.50**	**9.50**	**9.10**	**33.50**	**29.80**	610
Llama-3.2-1B	LoCoMo	12.86	10.50	11.25	9.18	7.38	6.82	11.90	10.38	51.89	48.27	16,910
	ReadAgent	7.75	6.03	5.96	5.12	1.93	2.30	12.46	11.17	44.64	40.15	665
	MemoryBank	17.30	14.03	13.18	10.03	7.61	6.27	15.78	12.94	52.61	47.53	274
	MemGPT	10.16	7.68	9.19	6.96	4.02	4.79	11.14	8.24	49.75	45.11	16,950
	A-MEM	28.51	24.13	19.06	11.71	17.80	10.28	17.55	14.67	58.81	54.28	1,376
	LightMem	**30.80**	**26.20**	**22.40**	**14.20**	**19.90**	**12.10**	**18.90**	**16.10**	**63.50**	**58.40**	750
Llama-3.2-3B	LoCoMo	8.37	6.93	6.88	5.77	4.37	4.40	10.65	9.29	30.25	28.46	16,910
	ReadAgent	3.25	2.51	2.47	1.78	3.01	3.01	5.57	5.22	15.78	14.01	461
	MemoryBank	7.61	6.03	6.19	4.47	3.49	3.13	4.07	4.57	18.65	17.05	263
	MemGPT	4.32	3.51	5.32	3.99	2.68	2.72	5.64	5.54	21.45	19.37	16,956
	A-MEM	28.14	23.87	17.44	11.74	26.38	19.50	12.53	11.83	42.04	40.60	1,126
	LightMem	**30.20**	**25.90**	**20.10**	**13.50**	**28.20**	**21.10**	**14.20**	**13.50**	**47.10**	**44.80**	640

4.2 Performance

Performance. As shown in Table 2, *LightMem* achieves the best overall performance on the LoCoMo benchmark. In comparison to baselines that directly rely on long-context replay (such as LoCoMo and MemGPT), *LightMem* demonstrates more stable performance across the majority of question categories. It has particularly distinct advantages for multi-hop and temporal questions, all while not relying on a single long-context input. In comparison with summarization- or compression-based methods (MemoryBank and ReadAgent), *LightMem* consistently outperforms them across all categories, with larger margins on multi-hop and temporal reasoning. When compared to the strongest memory-centric baseline, A-MEM, *LightMem* further improves overall performance in most model settings. On GPT-4o, *LightMem* attains an F1 score of 34.50 for multi-hop questions, surpassing A-MEM's score of 32.86. Overall, *LightMem* demonstrates more robust long-term memory across different model scales and question types. In addition, it maintains significant advantages in cross-session reasoning tasks.

Table 3 reports the comparison on DialSim with GPT-4o-mini. We incorporate METEOR and SBERT similarity. This is driven by the characteristics of long-term dialogue generation. In long-term dialogue generation, semantically ac-

curate answers can be presented in diverse surface forms. Consequently, metrics based solely on n-grams may undervalue advancements. METEOR partially accounts for morphological and synonym-level matching, while SBERT directly measures semantic alignment between the generated response and the reference, providing a complementary view of memory effectiveness. Across all metrics, *LightMem* achieves the best results. Importantly, *LightMem* improves not only lexical overlap (e.g., ROUGE-L/ROUGE-2 and METEOR) but also semantic similarity, with SBERT increasing from 19.51 (A-MEM) to 23.40. This indicates that the gains are not limited to reproducing similar wording, but also reflect stronger semantic consistency enabled by more effective use of long-term conversational history.

Table 3: Comparison of different memory mechanisms on DialSim with GPT-4o-mini.

Method	F1	BLEU-1	ROUGE-L	ROUGE-2	METEOR	SBERT
LoCoMo	2.55	3.13	2.75	0.90	1.64	15.76
MemGPT	1.18	1.07	0.96	0.42	0.95	8.54
A-MEM	3.45	3.37	3.54	3.60	2.05	19.51
LightMem	**4.12**	**3.95**	**4.20**	**4.15**	**2.48**	**23.40**

Generalization. We evaluate all methods across diverse backbones, ranging from large models (GPT-4o / GPT-4o-mini) to small open-source models (Qwen2.5 and Llama-3.2). As shown in Table 2, *LightMem* remains the best or near-best method across these settings, indicating that its gains are robust and not tied to a specific backbone.

4.3 Ablation Study

To quantify the contribution of each component in *LightMem*, we conduct ablation studies by removing or simplifying one key module at a time while keeping the rest of the system unchanged. We consider the following variants:

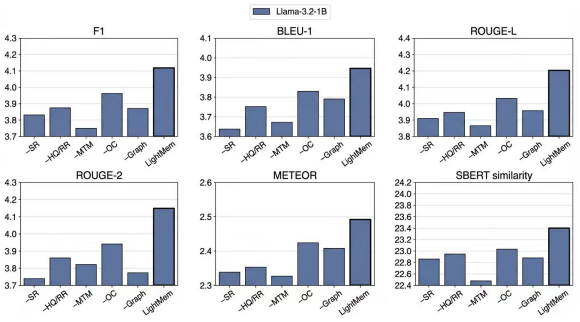
- **w/o semantic reranking** We remove the LM-based semantic filtering stage and keep only embedding-based Top-K retrieval.
- **w/o HQ and retrieval routing** We disable HQs as well as the controller-based routing/budget allocation, and retrieve using the original query directly.
- **w/o MTM** We remove the MTM layer, relying only on the current context (STM) and LTM for retrieval and generation.
- **w/o offline consolidation** We disable the offline consolidation pipeline so that LTM is not periodically updated from MTM.
- **w/o graph structure** We replace the graph-structured LTM with a flat vector store.

Figure 3 shows that removing any single component consistently degrades performance across metrics. Disabling semantic reranking or HQ-based retrieval routing leads to clear drops in F1; on Llama-3.2-1B, *LightMem* achieves an F1 of 4.12, while the variant without semantic reranking scores 3.83 and the variant without HQ and retrieval routing scores 3.87. Removing MTM further lowers performance, with an F1 of 3.75, indicating the benefit of retaining mid-term episodic memory. In comparison, disabling offline consolidation results in a smaller but still consistent decline, with an F1 of 3.96 on Llama-3.2-1B. Removing the graph structure also reduces both lexical and semantic quality, as reflected by a lower SBERT similarity of 22.82 compared to 23.40 for the full *LightMem*. Overall, the full *LightMem* achieves the best results, suggesting that these components contribute complementary benefits.

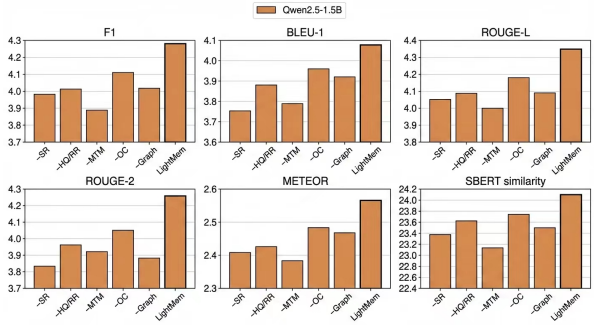
4.4 Scalability and Latency

Under a unified deployment configuration, we assess the scalability and latency by utilizing GPT-4o-mini. Subsequently, we present the following latency metrics:

- **Retrieval Latency (ms).** Time from receiving a user query to finishing memory retrieval and constructing the final prompt. It includes local computation but excludes remote GPT-4o-mini generation time. For the full-context LoCoMo baseline, retrieval latency is 0 ms since no external memory is retrieved.
- **End-to-End Latency (ms).** The time from receiving a user query to producing the final response, including retrieval, remote API transmission and queuing, and GPT-4o-mini inference and decoding.
- **P50 / P95.** The median (P50) and 95th-percentile (P95) latency across all samples. P95 captures tail latency from network jitter and service queuing, reflecting real-world system stability.



(a) Ablation study on Dialsim with Llama-3.2-1B



(b) Ablation study on Dialsim with Qwen2.5-1.5B

Figure 3: Ablation study on DialSim. We report F1, BLEU-1, ROUGE-L, ROUGE-2, METEOR, and SBERT similarity.

Table 4: Scalability and latency analysis across five baselines on GPT-4o-mini.

Method	Retrieval Latency P50 (ms)↓	Retrieval Latency P95 (ms)↓	End-to-End P50 (ms)↓	End-to-End P95 (ms)↓
LoCoMo	0	0	2054	3658
ReadAgent	34	97	614	1027
MemoryBank	16	51	439	1047
MemGPT	143	451	2087	3451
A-MEM	856	1583	914	3682
LightMem	83	167	581	1325

Table 4 compares retrieval and end-to-end latency on GPT-4o-mini. Among memory-enabled methods, *LightMem* achieves a good balance between retrieval overhead and end-to-end responsiveness. Its retrieval latency remains low, with P50/P95 of 83/167 ms, substantially lower than MemGPT and especially A-MEM, which exhibits heavy tail latency. *LightMem* also shows competitive end-to-end latency, with a P50 of 581 ms and a controlled P95, indicating stable behavior under remote API settings. When considered together with the effective context length in Table 2, these results are more interpretable: LoCoMo and MemGPT rely on replaying large dialogue histories with effective contexts close to 16K tokens, whereas *LightMem* consistently operates with much shorter contexts (around 1K tokens on GPT-4o-mini). This reduction in effective context length lowers prompt construction and inference cost, contributing to better scalability and practical deployability.

4.5 Performance Stability under MTM Growth

We analyze how MTM size affects performance stability on DialSim with Llama-3.2-1B. Rather than rerunning the model with different memory sizes, we report cumulative F1 at several natural growth stages along the same full dialogue trajectory, approximately at 100, 1,000, 5,000, and 10,000 MTM entries. As shown in Table 5, the gap between the two methods is small when MTM remains relatively small. However, as memory entries accumulate along the dialogue trajectory, pure vector retrieval becomes increasingly affected by retrieval noise, whereas *LightMem* maintains more stable performance. This trend is consistent with the role of Stage 2 semantic filtering, which helps preserve retrieval quality as the memory store grows.

Table 5: Performance stability under natural MTM growth on DialSim with Llama-3.2-1B. Statistics are computed cumulatively along the same full dialogue trajectory at different MTM sizes.

MTM size	Vector retrieval	LightMem	ΔF1
∼100	3.95	3.98	0.03
∼1,000	3.90	4.05	0.15
∼5,000	3.86	4.09	0.23
10,000	**3.83**	**4.12**	**0.29**

4.6 Error Injection Stress Test

To study error accumulation in long-horizon memory systems, we conduct an end-to-end error injection stress test on DialSim under a fixed final Top-K memory budget. Group A is the full *LightMem* system. Group B injects 50% irrelevant HQs to simulate SLM-1 failures in query planning. Group C removes SLM-2 semantic reranking and directly

Table 6: Error injection stress test on DialSim under a fixed final Top-K memory budget.

Group	Setting	F1	SBERT
A	Full LightMem	**4.12**	**23.40**
B	SLM-1 50% HQ noise	3.95	23.08
C	w/o SLM-2 reranking	3.83	22.56
D	SLM-3 50% write noise	3.78	22.45
E	Cascading failure	1.85	11.20

uses the stage-one vector Top-K. Group D replaces 50% of SLM-3 writes with noisy strings, simulating persistent MTM pollution. Group E combines all three perturbations to simulate cascading failure. As shown in Table 6, performance degrades consistently as errors are introduced into different stages of the pipeline. Query noise in SLM-1 causes a moderate drop, suggesting partial robustness to retrieval-side perturbations. Removing SLM-2 leads to a larger decline, confirming the importance of semantic filtering after coarse retrieval. Noisy writes in SLM-3 further degrade performance, indicating that memory pollution introduces persistent downstream effects by contaminating future retrieval. Cascading failure causes the largest collapse, showing that errors can accumulate and amplify across turns when noisy retrieval, missing filtering, and corrupted writing occur simultaneously.

4.7 Update Gap Stress Test

As shown in Table 7, concurrent retrieval from MTM and LTM is important for handling the update gap. The full *LightMem* system achieves the best performance, with a multi-hop F1 of 28.85, whereas using only LTM or only MTM results in clear performance drops. This suggests that concurrent routing effectively bridges recent unconsolidated facts in MTM and older consolidated knowledge in LTM. A more challenging case appears in Group D. Under MTM noise saturation, the multi-hop F1 decreases to 23.10, indicating that heavy recent noise can interfere with retrieval during the update gap. This is likely because the fixed Top-K budget is increasingly occupied by irrelevant recent records, making the newest relevant fact harder to retrieve before it is consolidated into LTM. Overall, the results show that concurrent retrieval provides a practical and effective solution for update-gap reasoning, especially when recent and long-term evidence must be combined.

Table 7: Update gap stress test on the LoCoMo multi-hop subset with GPT-4o-mini.

Group	Setting	Multi-hop F1
A	Full LightMem (normal gap)	**28.85**
B	LTM-only retrieval	19.45
C	MTM-only retrieval	20.12
D	MTM noise saturation	23.10

5 Conclusion

We present *LightMem*, a lightweight external memory system for LLM agents driven by SLMs. *LightMem* decouples high-frequency online memory operations (query control, retrieval, and writing) from offline consolidation, and organizes memory into STM/MTM/LTM with user-scoped isolation. Online, it operates under a fixed retrieval budget and uses HQ-based routing plus a two-stage retrieval (coarse vector search followed by semantic consistency re-ranking) to reduce retrieval noise. Experiments on LoCoMo and DialSim show consistent gains across model scales, including +2.5 average F1 on LoCoMo, improved semantic consistency on DialSim, and low median latency (83 ms retrieval; 581 ms end-to-end).

6 Limitations

This work focuses on a specific design of online–offline decoupled memory pipelines driven by specialized small language models. The impact of alternative consolidation strategies and control policies is not fully explored and remains an interesting direction for future investigation.

7 Acknowledgements

This work was partially supported by the National Natural Science Foundation of China under grant 62572104, and 62220106008.

References

A Asai, Z Wu, Y Wang, A Sil, and H Self-RAG Hajishirzi. Learning to retrieve, generate, and critique through self-reflection. arxiv 2023. *arXiv preprint arXiv:2310.11511*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela

Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Haixia Han, Jiaqing Liang, Jie Shi, Qianyu He, and Yanghua Xiao. 2024. Small language model can self-correct. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18162–18170.

Kostas Hatalis, Despina Christou, Joshua Myers, Steven Jones, Keith Lambert, Adam Amos-Binks, Zohreh Dannenhauer, and Dustin Dannenhauer. 2023. Memory matters: The need to improve long-term memory in llm-agents. In *Proceedings of the AAAI Symposium Series*, volume 2, pages 277–280.

Chuanyang Hong and Qingyun He. 2025. Enhancing memory retrieval in generative agents through llm-trained cross attention networks. *Frontiers in Psychology*, 16:1591618.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. 2025. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32779–32798.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jiho Kim, Woosog Chay, Hyeonji Hwang, Daeun Kyung, Hyunseung Chung, Eunbyeol Cho, Yohan Jo, and Edward Choi. 2024. Dialsim: A real-time simulator for evaluating long-term multi-party dialogue understanding of conversation systems. *arXiv preprint arXiv:2406.13144*.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John F. Canny, and Ian Fischer. 2024. A human-inspired reading agent with gist memory of very long contexts. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.

Guangliang Liu, Zhiyu Xue, Xitong Zhang, Rongrong Wang, and Kristen Johnson. 2025. Smaller large language models can do moral self-correction. In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pages 56–65.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 2: short papers)*, pages 1773–1781.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 13851–13870. Association for Computational Linguistics.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: Towards llms as operating systems.

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.

Neelabh Sinha, Vinija Jain, and Aman Chadha. 2025. Are small language models ready to compete with large language models for practical applications? In *Proceedings of the 5th Workshop on Trustworthy NLP (TrustNLP 2025)*, pages 365–398.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. 2025. Membench: Towards more comprehensive evaluation on the memory of llm-based agents. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 19336–19352. Association for Computational Linguistics.

Xudong Wang, Chaoning Zhang, Jiaquan Zhang, Chenghao Li, Qigan Sun, Sung-Ho Bae, Peng Wang, Ning Xie, Jie Zou, Yang Yang, and 1 others. 2026. Efficient and interpretable multi-agent llm routing via ant colony optimization. *arXiv preprint arXiv:2603.12933*.

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

Jiaquan Zhang, Qigan Sun, Chaoning Zhang, Xudong Wang, Zhenzhen Huang, Yitian Zhou, Pengcheng Zheng, Chi lok Andy Tai, Sung-Ho Bae, Zeyu Ma, Caiyan Qin, Jinyu Guo, Yang Yang, and Hengtao Shen. 2026a. Tda-rc: Task-driven alignment for knowledge-based reasoning chains in large language models.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Yibei Liu, Chenghao Li, Qigan Sun, Shuai Yuan, Fachrina Dewi Puspitasari, Dongshen Han, Guoqing Wang, and 1 others. 2026b. Text summarization via global structure awareness. *arXiv preprint arXiv:2602.09821*.

Jiaquan Zhang, Chaoning Zhang, Shuxu Chen, Xudong Wang, Zhenzhen Huang, Pengcheng Zheng, Shuai

Yuan, Sheng Zheng, Qigan Sun, Jie Zou, and 1 others. 2026c. Learning global hypothesis space for enhancing synergistic reasoning chain. *arXiv preprint arXiv:2602.09794*.

Malu Zhang, Wenjie Wei, Zijian Zhou, Wanlong Liu, Jie Zhang, Ammar Belatreche, and Yang Yang. 2025a. Spike-driven lightweight large language model with evolutionary computation. *IEEE Transactions on Evolutionary Computation*.

Qinyao Zhang, Bin Guo, Yao Jing, Yan Liu, and Zhiwen Yu. 2024. Mindmemory: Augmented llm with long-term memory and mental personality. In *CCF Conference on Computer Supported Cooperative Work and Social Computing*, pages 462–476. Springer.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025b. A survey on the memory mechanism of large language model-based agents. *ACM Trans. Inf. Syst.*, 43(6):155:1–155:47.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, Xiaocheng Yang, Shuyi Guo, Daisy Zhe Wang, Zhenhailong Wang, Cheng Qian, Robert Tang, Heng Ji, and 1 others. 2025. Multiagentbench: Evaluating the collaboration and competition of llm agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8580–8622.

8 Appendices

8.1 Complete Algorithm and Process Specifications

8.1.1 Symbols and Data Structures

At interaction turn t, we denote the user input by x_t and the system response by y_t. The recent dialogue context maintained in short-term memory (STM) is represented by C_t, which corresponds to a truncated window of the latest conversation history. For each user u, the associated mid-term memory (MTM) store is denoted by M_u^{MTM}, while the global long-term memory (LTM), shared across users, is denoted by M^{LTM}. We use K to denote the fixed Top-K retrieval budget and B to denote the capacity limit of the MTM store. In addition, HQs refers to the hypotheses or query hints generated by SLM-1 for guiding memory retrieval, and ϕ_t denotes the metadata constraints applied at turn t, including user identifiers, temporal windows, and type tags.

Mid-Term Memory (MTM). Each MTM entry represents a compressed interaction record, consisting of a concise semantic summary, a retrieval embedding, associated temporal metadata (e.g., timestamps and access frequencies), and a user identifier for user-level isolation.

Long-Term Memory (LTM). Each LTM node represents a de-identified semantic knowledge unit and is stored in a lightweight graph structure, which supports associative organization and multi-hop access across memory units.

8.1.2 Online Retrieval Procedure

Algorithm 1 describes the full online retrieval process executed at each dialogue turn.

Algorithm 1 LightMem Online Retrieval

Require: User query x_t, truncated context C_t, user memory M_u^{MTM}, global memory M^{LTM}, retrieval budget K
Ensure: Retrieved memory set R_t
Intent Modeling and Query Control (SLM-1)
Infer high-level intent signals from (x_t, C_t), including (i) degree of personalization and (ii) reliance on recent vs. long-term information.
Generate a set of HQs $\{q_t^{(i)}\}_{i=1}^{n}$.
Produce metadata constraints ϕ_t (e.g., user identifier and optional temporal filters).
Allocate retrieval quotas across memory layers under a fixed Top-K constraint.
Stage 1: Metadata-Constrained Coarse Retrieval
Initialize candidate set $C \leftarrow \emptyset$.
for each $q_t^{(i)}$, $i = 1, \ldots, n$ **do**
    Retrieve candidates from M_u^{MTM} and/or M^{LTM} using vector search under ϕ_t.
    Let $K_1^{(i)} \leftarrow \lceil \frac{2K}{n} \rceil$ and collect up to $K_1^{(i)}$ candidates as $C^{(i)}$.
    Update $C \leftarrow C \cup C^{(i)}$.
end for
Optionally truncate C to size $2K$ if $|C| > 2K$.
Stage 2: Semantic Filtering and Compression (SLM-2)
Perform semantic consistency checking and relevance assessment conditioned on $\{q_t^{(i)}\}$ and C.
Select up to K items to form the final retrieved set $R_t \subseteq C$ with $|R_t| \leq K$.
Output
return R_t (to be appended to the prompt for response generation).

8.1.3 Response Generation

At each dialogue turn, the retrieved memory set R_t is concatenated with the short-term context C_t and provided to a fixed response generator (e.g., GPT-4o-mini) to produce the system response y_t. This stage does not involve any memory control or modification, and the same response generation setup is used for *LightMem* and all baseline methods in our experiments.

8.1.4 Online Memory Writing and MTM Maintenance

After response generation, *LightMem* updates mid-term memory via SLM-3, as described in Algorithm 2.

Algorithm 2 LightMem Online Memory Writing and MTM Maintenance

Require: Current interaction (x_t, y_t, C_t), user mid-term memory M_u^{MTM}, capacity bound B
Ensure: Updated M_u^{MTM}
Memory Extraction and Compression (SLM-3)
Extract reusable, user-relevant information from the current interaction (x_t, y_t, C_t).
Compress the extracted content into a concise semantic memory summary.
Memory Appending
Append the compressed memory summary to the user-scoped MTM M_u^{MTM}.
Redundancy and Conflict Handling
Identify highly repetitive or semantically overlapping MTM entries and merge or rewrite them to reduce redundancy.
Resolve conflicting information using temporal cues and accumulated evidence strength.
Capacity-Bound Maintenance
if $|M_u^{\mathrm{MTM}}| > B$ **then**
    Evict stale or low-utility entries based on recency and access statistics.
    Further compress redundant content to enforce the capacity bound.
end if
return Updated M_u^{MTM}.

8.1.5 Offline Consolidation into Long-Term Memory

The consolidation procedure is summarized in Algorithm 3.

8.1.6 Online-Offline Decoupling Guarantee

All operations in Algorithm 1 and Algorithm 2 are executed under strict latency and compute constraints using small language models. In contrast, Algorithm 3 is performed asynchronously and does not block online interaction. This design ensures that (i) the cost of online retrieval remains bounded by $O(K)$, and (ii) the evolution of long-term memory does not introduce additional overhead to online inference.

Algorithm 3 LightMem Offline Consolidation into LTM

Require: MTM items flagged for consolidation, existing LTM graph M^{LTM}
Ensure: Updated LTM graph M^{LTM}
Selection of Candidate Episodes
Select MTM items that are newly written, frequently retrieved, or marked as low-utility under MTM capacity pressure.
Abstraction and De-identification
Abstract episodic interaction records into user-agnostic semantic knowledge units.
Remove personal identifiers and session-specific details.
Graph Insertion and Update
Perform similarity search over M^{LTM} to identify nearby semantic anchors.
Insert new nodes and edges within the local neighborhood of matched anchors.
Merge or update existing nodes when semantic overlap exceeds a threshold.
Evidence Accumulation and Forgetting
Aggregate supporting evidence across consolidation cycles.
Apply confidence decay to weakly supported knowledge units.
Remove or down-weight stale or incidental knowledge.
return Updated M^{LTM}.

9 Experiment

9.1 Statistical Significance Analysis

We report mean $\pm$ standard deviation across three random seeds for all stochastic components (see Table 8). Statistical significance is assessed using paired bootstrap resampling over evaluation instances (1,000 resamples), reporting the mean difference (Δ), its 95% confidence interval (CI), and two-sided p-values. We present results against the strongest baseline (A-MEM) under the same backbone model and retrieval budget.

Table 8: Statistical significance results against A-MEM under the same backbone and retrieval budget.

Category	Method	F1 (mean $\pm$ std)	Δ F1	95% CI of Δ	p-value
Multi-hop	A-MEM	27.02 $\pm$ 0.31	—	—	—
	LightMem	**28.85** $\pm$ **0.28**	1.83	[+1.24, +2.41]	**0.001**
Temporal	A-MEM	45.85 $\pm$ 0.37	—	—	—
	LightMem	**46.20** $\pm$ **0.34**	0.35	[+0.12, +0.58]	**0.008**

9.2 Failure Analysis: Ambiguous Intent Decomposition in SLM-1

We observe a mild failure mode when SLM-1 generates both episodic and general HQs for an underspecified recall query. For example, given the user query "Can you remind me what I decided about the project timeline?", the relevant decision is stored in MTM, while general planning advice may exist in LTM. SLM-1 may produce HQs for (i) decision recall (MTM), (ii) project disambiguation (MTM), and (iii) generic timeline principles (LTM). Under a fixed Top-K budget, allocating quota to the LTM HQ can introduce a small amount of generic content in the retrieved set. In practice, the final response typically remains correct (the key MTM decision is recalled), but becomes slightly less focused or concise due to additional general advice.

This "focus dilution" mainly affects lexical-overlap metrics (e.g., F1/BLEU), while semantic similarity remains stable, and SLM-2's semantic filtering helps preserve the most relevant MTM evidence.

10 Full Prompt Templates and Control Interface Definitions

This appendix provides the complete prompt templates used by all language model components in *LightMem*, including SLM-1, SLM-2, SLM-3, and the offline LLM. All prompts are designed to be functionally complete, explicitly specifying model roles, accessible inputs, required tasks, prohibited behaviors, and output formats, to ensure reproducibility and auditability of the system implementation.

10.1 Unified Prompt Design Principles

All prompts in *LightMem* adhere to a unified design schema in order to eliminate implicit assumptions and avoid relying on undeclared capabilities. Specifically, each prompt explicitly defines the **Role** of the model, namely its functional responsibility within the overall system; the **Input**, namely the information made accessible to the model; the **Task**, namely the concrete objectives the model is required to accomplish; the **Constraints**, namely the behaviors or actions the model is prohibited from performing; and the **Output Format**, namely a fixed and machine-parsable specification for its returned results. This design ensures that online models are used for structured control and decision-making rather than open-ended generation.

10.1.1 Full Prompt Template (SLM-1)

Role SLM-1 serves as the query decomposition and routing engine within the memory system. Its function is not to answer the user directly, but to convert the user's raw input into a set of explicit HQs that can be used to retrieve the required information from the memory database.

Input The model receives two forms of input: the current user query and a truncated dialogue context from the ongoing session.

Task Logic For each user input, SLM-1 first identifies missing or underspecified information in the request. This includes vague or ambiguous expressions, unresolved pronouns such as "it," "that," or "the project," implicit dependencies on prior conversational context, and incomplete temporal references such as "recently," "last time," or "before." It then rewrites the original request into one or more independent and explicit HQs suitable for downstream retrieval. These queries should not simply copy the user's raw wording, but instead resolve implicit entities, make hidden constraints explicit, and normalize the request into retrieval-oriented formulations. When a request simultaneously involves user-specific preferences and objective factual information, the two aspects should be separated into different queries. Likewise, temporal or spatial constraints should be explicitly formulated whenever they are implied by context. Finally, for each generated HQ, SLM-1 assigns a target memory layer according to the nature of the information need. Queries involving user-specific or personalized information should be routed to MTM, whereas queries involving public, factual, or generally shared knowledge should be routed to LTM.

Constraints SLM-1 must not answer the user, must not generate user-facing natural language responses, and must not perform retrieval itself.

Output Format The output must be returned in JSON format only.

10.2 SLM-2: Semantic Consistency Filtering and Candidate Compression

SLM-2 is responsible for semantic consistency checking and candidate selection over a fixed-size set of retrieved memory items, with the goal of suppressing retrieval noise and reducing the effective context size. Its role is purely selective: it neither generates new content nor rewrites existing memory entries.

10.2.1 Full Prompt Template (SLM-2)

Role SLM-2 serves as the semantic consistency filtering engine for memory retrieval. Its function is not to answer the user or produce any new content, but to determine which retrieved memory items should be retained for downstream use.

Input The model takes as input a set of HQs together with a fixed-size list of candidate memory summaries and their associated metadata.

Task For each candidate memory item, SLM-2 evaluates whether the item is semantically consistent with at least one HQ. Based on this assessment, it selects the most relevant memory items subject to a fixed budget constraint. The objective is to pre-

serve only those items that meaningfully support the retrieval intent expressed by the HQ set.

Guidelines Relevance should be determined primarily on the basis of semantic alignment rather than superficial lexical overlap. In addition, each selected memory item should provide direct support for at least one HQ, rather than being merely topically related in a loose sense.

Constraints SLM-2 must not rewrite or modify memory content, must not generate explanations or user-facing responses, and must not perform retrieval or expansion. Its function is restricted to selection only.

Output Format The output must be returned in JSON format only.

10.3 SLM-3: Interaction Summarization and Online MTM Maintenance

SLM-3 is responsible for compressing the current interaction into reusable MTM entries and for supporting online maintenance of the MTM store.

10.3.1 Full Prompt Template (SLM-3)

Role SLM-3 serves as the memory writing and maintenance engine for MTM. Its function is to distill interaction-level information into compact semantic memory units that can be reused in future user interactions.

Input The model receives as input the current user utterance, the generated system response, and the relevant short-term dialogue context associated with the ongoing interaction.

Task SLM-3 is responsible for identifying user-relevant information that may be useful beyond the current turn, compressing that information into concise semantic memory entries, and supporting the incremental maintenance of the MTM store over time. Its purpose is not merely to record interaction content, but to transform transient dialogue into reusable memory representations.

Guidelines The model should focus on information that is likely to remain useful in future interactions. It should prefer concise, self-contained semantic summaries rather than preserving raw dialogue content or turn-level transcripts.

Constraints SLM-3 must not store full dialogue transcripts, must not retrieve or search existing memory, must not perform cross-user abstraction, and must not generate responses intended for the user.

Output Format The output should consist of one or more short semantic memory summaries.

10.4 Offline LLM: Long-Term Memory Consolidation

Role The offline LLM serves as the consolidation model responsible for maintaining LTM. Its role is to transform selected MTM content into stable long-term knowledge representations and to integrate those representations into the global memory structure.

Input The model receives as input a batch of MTM items selected for consolidation. These items may include newly written entries, retrieval-reactivated entries, and low-utility entries that have been flagged under MTM capacity pressure.

Task The model is responsible for abstracting episodic MTM items into de-identified semantic knowledge units, identifying semantic overlap between candidate knowledge units and existing LTM nodes, and determining whether each candidate should be merged with existing knowledge, used to update an existing node, or discarded. For retained knowledge, it further integrates the resulting representations into the LTM graph by inserting new nodes and edges or updating existing ones.

Guidelines During abstraction, the model should remove user-specific and session-specific details so that the resulting knowledge remains de-identified and reusable across contexts. It should prioritize stable and generalizable knowledge over episodic, incidental, or overly context-bound information.

Constraints The offline LLM must not generate user-facing responses and must not directly modify MTM. In addition, this process is executed offline and asynchronously, and therefore must not affect the latency of online interactions.

Output Format The output should consist of a set of newly inserted or updated LTM knowledge units together with their associated graph links.

11 LTM Graph Schema and Maintenance Dynamics

This section supplements Section 3.6 by providing a detailed definition of the schema used for the

graph-structured LTM and by discussing its maintenance dynamics, including update frequency, node growth behavior, and the impact of offline consolidation on downstream reasoning performance.

11.1 Graph Schema: Node and Edge Types

As described in Section 3.6, LTM is designed as a lightweight, de-identified semantic knowledge graph. Unlike MTM, which stores episodic interaction summaries, LTM aims to capture stable factual knowledge and domain-level regularities. To support cross-task generalization and multi-hop reasoning, elements in the LTM graph are standardized into a small set of node and edge types, summarized in Table 9.

11.2 Maintenance Dynamics and Performance Impact

LightMem adopts a decoupled architecture that separates online retrieval from offline consolidation. The offline LLM processes updates in incremental batches, ensuring that maintenance does not block real-time user interaction. Table 10 summarizes key operational metrics of the consolidation process and analyzes their impact on model performance.

Table 9: Node and edge types used in the LTM knowledge graph.

Category	Type	Description and Example
Nodes	Entity	Concrete objects, locations, or named items extracted from interactions (e.g., "Python script", "Paris", "Project X").
	Concept	Abstract categories or generalized classes that characterize shared properties of entities (e.g., "Programming Language", "Capital City", "Urgent Task").
Edges	IsA	Hierarchical membership relation between an entity and a concept (e.g., *Paris* $\xrightarrow{\text{IsA}}$ *Capital City*).
	HasProperty	Links an entity to a specific attribute or state (e.g., *Project X* $\xrightarrow{\text{HasProperty}}$ *Completed*).
	RelatedTo	Generic semantic association based on co-occurrence or functional relatedness (e.g., *SNN* $\xrightarrow{\text{RelatedTo}}$ *Neural Operator*).
	Implies	Logical or causal dependency extracted from reasoning traces (e.g., *High Density* $\xrightarrow{\text{Implies}}$ *Congestion*).

Table 10: LTM update frequency, consolidation cost, and ablation results on reasoning accuracy.

Metric	Value	Description and Analysis
Batch update interval	Every 10–15 turns	Offline consolidation is periodically triggered when MTM accumulates sufficient new entries or reaches capacity pressure, enabling amortized long-term updates without interfering with online processing.
Node growth rate	$\sim$1 node / 4 turns	On average, only one new semantic node is added to LTM every four dialogue turns. This sparsity reflects the high compression ratio of *LightMem*, where noise is filtered and only high-value evidence is retained for persistent storage.
Offline processing time	$\sim$3.5 s / batch	Average time for the offline LLM to perform abstraction, graph linkage, and structural updates. This latency occurs entirely on the offline path and is fully decoupled from online retrieval (average retrieval latency $\sim$83 ms), making it invisible to users.
Reasoning accuracy (F1)	4.12 (full) vs. 3.96 (no update)	The full *LightMem* system (4.12) outperforms a variant without offline consolidation (3.96), corresponding to an approximately 4% performance drop when LTM evolution is disabled. This result highlights the importance of sustained LTM growth and consolidation for long-horizon reasoning.