MetaMem: Evolving Meta-Memory for Knowledge Utilization through Self-Reflective Symbolic Optimization

Anonymous ACL submission

Abstract

Existing memory systems enable Large Language Models (LLMs) to support long-horizon human-LLM interactions by persisting historical interactions beyond limited context windows. However, while recent approaches have succeeded in constructing effective memories, they often disrupt the inherent logical and temporal relationships within interaction sessions, resulting in fragmented memory units and degraded reasoning performance. In this paper, we propose MetaMem, a novel framework that augments memory systems with a self-evolving meta-memory, aiming to teach LLMs how to effectively utilize memorized knowledge. During meta-memory optimization, MetaMem iteratively distills transferable knowledge utilization experiences across different tasks by self-reflecting on reasoning processes and performing actions to update the current meta-memory state. The accumulated meta-memory units serve as explicit knowledge utilization experiences, guiding the LLM to systematically identify and integrate critical evidence from scattered memory fragments. Extensive experiments demonstrate the effectiveness of MetaMem, which significantly outperforms strong baselines by over 3.6%. All code and datasets will be released on GitHub.

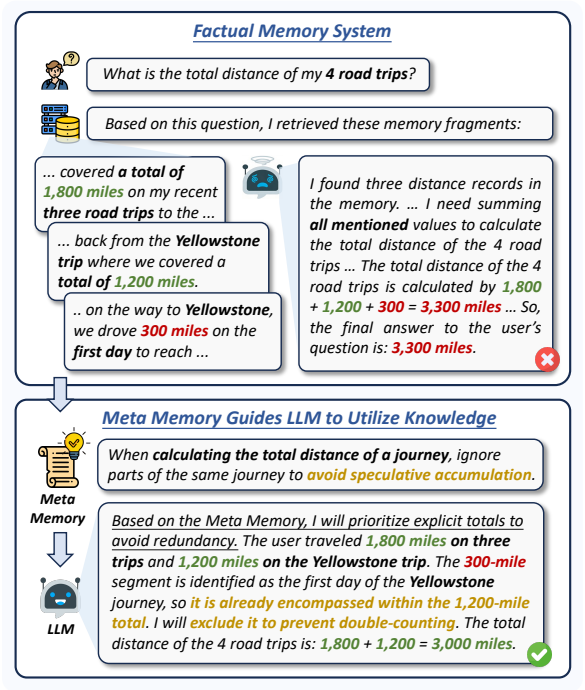


Figure 1: Illustration of Our MetaMem Framework. **Green** indicates correct evidence, while **red** represents distracting evidence. MetaMem guides the LLM to effectively utilize knowledge from scattered memory fragments, thereby generating the correct answer.

1 Introduction

Large Language Models (LLMs) (Liu et al., 2024a; Yang et al., 2025; Team et al., 2025) have demonstrated remarkable capabilities in complex tasks (Lewis et al., 2020; He et al., 2021; Liu et al., 2025b), largely attributed to their emergent abilities. Nevertheless, LLMs are usually constrained by limited context window sizes (Wang et al., 2024; Liu et al., 2025a), which restricts their effectiveness in handling long-horizon historical human-LLM interaction scenarios. To address this limitation, researchers have developed external memory systems to persist historical context beyond the model's context window (Packer et al., 2023; Kang et al., 2025). These systems typically process interaction sessions to extract and store factual knowledge or user preferences as manageable memory units, maintaining long-term continuity through mechanisms such as updating and forgetting (Zhong et al., 2024). However, interrelated evidence is often fragmented across dispersed sessions, which hinders effective information utilization.

To mitigate this issue, recent studies have focused on associating related information distributed across different human-LLM interaction sessions. These approaches (Fang et al., 2025; Xu et al., 2025) consolidate related knowledge to construct more effective and comprehensive memo-

ries. Specifically, they achieve this by topic-based clustering (Fang et al., 2025) or by transforming interaction sessions into structured memory fragments (Xu et al., 2025). However, the resulting memory units are often fragmented or even contradictory, which disrupts the original logical and temporal relationships among facts. Such memory construction prevents LLMs from fully leveraging the accumulated knowledge.

As illustrated in Figure 1, for the given query, related memories may contain several inconsistent versions of factual knowledge regarding the same event, such as different estimates of a total travel distance of the "Yellowstone trip", including "1200 miles" and "300 miles". Due to the fragmented nature of these memory units, the model becomes confused, leading to incorrect answers. By introducing knowledge usage experiences, the model ignores misleading segments of the same journey, thereby enabling accurate reasoning. This observation highlights the crucial role of guiding LLMs to learn how to utilize memorized knowledge effectively, which aligns with the "learning to learn" principle in meta-learning (Flavell, 1979).

This paper proposes MetaMem, a novel memory system framework augmented with a self-evolving meta-memory for memorized knowledge utilization. During meta-memory evolving, MetaMem first generates diverse reasoning responses conditioned on the user query, memory fragments, and the current meta-memory experiences. It then reflects on these responses under the guidance of correctness-based rewards to distill generalizable insights, and subsequently proposes updates to optimize the meta-memory. After optimization, the meta-memory accumulates transferable knowledge utilization experiences, thereby guiding the LLM to more effectively and systematically exploit knowledge embedded in the memory fragments.

Our experimental results demonstrate the effectiveness of MetaMem by consistently outperforming all baseline models with more than 3.6% improvements. Notably, the advantages of MetaMem generalize across different knowledge usage systems, highlighting its strong generalization capability. Moreover, MetaMem effectively benefits from out-of-domain data through constructed meta-memory and exhibits substantial scalability with respect to the amount of training data. This property indicates significant potential for enabling MetaMem to continuously improve through long-term human-LLM interactions.

2 Related Work

Large Language Models (LLMs) (Liu et al., 2024a; Yang et al., 2025; Team et al., 2025) have demonstrated remarkable capabilities across a wide range of complex tasks (He et al., 2021; Trivedi et al., 2023; Liu et al., 2025b). Nevertheless, due to the limited context window of LLMs and the "lost-in-the-middle" phenomenon (Liu et al., 2024b), they face substantial difficulties when reasoning over long-horizon human–LLM interaction sessions. To mitigate these issues, existing approaches typically segment interaction histories into discrete chunks and retrieve relevant fragments to serve as model input (Zhou et al., 2025; Lewis et al., 2020). However, such strategies neglect the inherent coherence of interaction sessions, often leading to the retrieval of fragmented contexts that omit critical information necessary for accurately answering the query.

Recent studies have focused on developing external memory systems to store long-term memory, aiming to better manage and utilize historical interaction sessions (Packer et al., 2023; Park et al., 2023; Zhong et al., 2024). These methods typically treat historical interaction sessions as a manageable memory resource and systematically maintain long-term memory through hierarchical storage mechanisms (Kang et al., 2025) or continuous memory updating and forgetting (Zhong et al., 2024). Such designs enable LLMs to access more useful information across sessions and support long-horizon interactions. However, these methods lack an effective memory construction mechanism for systematically integrating information on similar topics that is scattered across different sessions, resulting in fragmented historical interaction memories.

To address this challenge, several studies have begun exploring effective memory construction methods, with a particular focus on extracting key information from the long context of human–LLM interactions. LightMem (Fang et al., 2025) constructs long-term memory from historical interactions through a memory construction pipeline that integrates text compression, similar-topic aggregation, and summarization to capture the most crucial information. G-Memory (Zhang et al., 2025) organizes historical interactions into graph structures to capture relationships across different topical sessions, enabling the model to better access key information. While these memory systems effectively construct memory for LLMs, they often disrupt the original logical and temporal relation-

ships among memory entries, preventing LLMs from fully exploiting the constructed memories. In contrast, MetaMem facilitates memory utilization by maintaining a meta-memory that conditions and coordinates different retrieved memories.

3 Methodology

In this section, we introduce MetaMem, a memory system framework augmented by a self-evolving meta-memory. We first introduce the preliminaries of meta-memory augmented memory systems (Sec. 3.1). Then, we detail the iterative optimizing process for meta-memory updation (Sec. 3.2).

3.1 The Architecture of Meta-Memory Augmented Memory System

To enable LLMs to effectively leverage long-horizon human-LLM interaction histories, existing methods (Kang et al., 2025; Fang et al., 2025) typically construct an explicit memory system. Such systems organize historical interactions into structured contextual memories by extracting and summarizing key information, thereby facilitating more accurate responses to a user query q.

Specifically, the memory system (Fang et al., 2025) first identifies sessions that share identical or highly similar topics from long-horizon human-LLM interaction data. For each distinct topic, the corresponding sessions are aggregated into a generalized memory representation m_i, which captures the salient information shared across those topic-aligned sessions. Through this process, a memory set $\mathcal{M} = \{m_1, \ldots, m_n\}$ covering diverse topics is constructed, where n denotes the number of topics. Finally, the long-term memory set $\mathcal{M}$ is provided to the LLM as additional context to assist in answering the user question q:

$$y = \text{LLM}(\text{Instruct}_{\text{Gen}}(q, \mathcal{M})), \tag{1}$$

where $\text{Instruct}_{\text{Gen}}$ denotes the instruction designed to prompt the LLM to generate the response y conditioned on the long-term memory $\mathcal{M}$. However, existing memory systems often disrupt the logical and temporal relationships within contextual memory during the chunking and summarization process. As a result, the memory unit m in $\mathcal{M}$, although containing key evidence, may not be effectively identified or exploited by the LLM.

In contrast to prior methods (Fang et al., 2025), we propose MetaMem, a meta-memory augmented system that enhances LLMs' ability to learn from and utilize external memory composed of key information. Specifically, MetaMem learns a task-agnostic meta-memory $\mathcal{E}$ by training on a multi-task dataset $\mathcal{D}$, enabling the model to acquire transferable experience in memory utilization and to rapidly adapt to different tasks:

$$\mathcal{E}_T = \text{MetaMem}(\mathcal{D}, \text{LLM}(\cdot)), \tag{2}$$

where the meta-memory $\mathcal{E}_T$ encapsulates learned experiences about how to effectively leverage memory across diverse tasks. The training dataset is formulated as $\mathcal{D} = \{(q_i, a_i, \mathcal{M}_i)\}_{i=1}^{N}$, where q_i denotes the question, a_i is the ground-truth answer, and $\mathcal{M}_i$ is the corresponding memory set for q_i. During inference, the optimized meta-memory $\mathcal{E}_T$ is employed to guide the LLM in more effectively utilizing the external memory set $\mathcal{M}$ to generate the response y:

$$y = \text{LLM}(\text{Instruct}_{\text{Gen}}(q, \mathcal{M}, \mathcal{E}_T)), \tag{3}$$

where $\text{Instruct}_{\text{Gen}}$ denotes the instruction template used to prompt the LLM to answer the question.

3.2 Meta-Memory Evolution for Learning to Use Memorized Contents

In this subsection, we describe the procedure for constructing the meta-memory with LLMs using the training dataset $\mathcal{D}$. We first present the overall workflow of meta-memory construction and then elaborate on the meta-memory update strategies based on self-reflection.

The Workflow of Meta-Memory Learning. To balance efficiency and effectiveness, MetaMem adopts a mini-batch learning strategy that partitions the training dataset $\mathcal{D}$ into multiple batches $\{\tilde{\mathcal{D}}_1, \ldots, \tilde{\mathcal{D}}_T\}$. During optimization, we initialize the meta-memory as an empty state $\mathcal{E}_0 = \emptyset$ and iteratively update it across batches. This iterative process continues until all batches are processed, yielding the final meta-memory $\mathcal{E}_T$.

At each optimization step t, the current meta-memory $\mathcal{E}_t = \{e_t^1, \ldots, e_t^n\}$ consists of multiple memory usage experiences, where each e_t^i denotes an individual meta-memory unit. We utilize the t-th data batch $\tilde{\mathcal{D}}_t$ to evolve $\mathcal{E}_t$ into an updated state $\mathcal{E}_{t+1}$. Formally, for each instance $(q, a, \mathcal{M})$ in the batch $\tilde{\mathcal{D}}_t$, we employ a self-reflection-based optimization function, denoted as `Update`, to generate a candidate meta-memory update operation $\tilde{O}_t$ based on the current meta-memory state $\mathcal{E}_t$ and the batch data at step t:

$$\tilde{O}_t = \texttt{Update}(\tilde{\mathcal{D}}_t, \mathcal{E}_t), \tag{4}$$

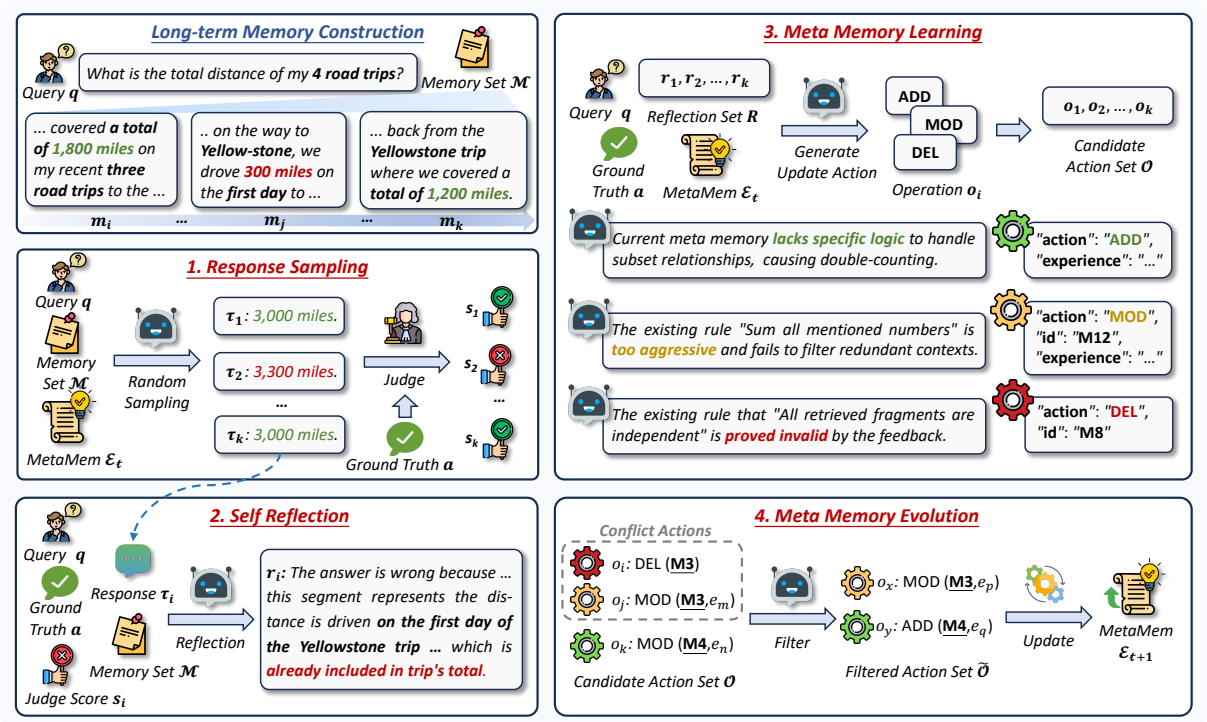


Figure 2: Overview of MetaMem Model. MetaMem evolves through environmental feedback, guiding the memory system to utilize factual knowledge through Meta Memory.

where $\tilde{O}_t = \{o_t^1, \ldots, o_t^m\}$ is a set of m update operations for refining the meta-memory $\mathcal{E}_t$ according to the performance of the meta-memory-augmented memory system. Each operation o_t^i belongs to one of the following three action types:

- *Addition*: o_t^i contains the identifier ADD that aims to append a newly generated meta-memory unit to $\mathcal{E}_t$.
- *Deletion*: o_t^i contains the identifier DEL and specifies the index j of an existing meta-memory unit $e_t^j \in \mathcal{E}_t$ to be removed. This operation is designed to eliminate obsolete or erroneous experience units.
- *Modification*: o_t^i includes the identifier MOD and the index j of the meta-memory unit $e_t^j \in \mathcal{E}_t$ to be updated, and replaces e_t^j with a modified unit e_{t+1}^j.

Finally, we apply the update operations $\tilde{O}_t$ to the current meta-memory $\mathcal{E}_t$ using the execution function Exec, producing the optimized meta-memory $\mathcal{E}_{t+1}$:

$$\mathcal{E}_{t+1} = \mathsf{Exec}(\tilde{O}_t, \mathcal{E}_t). \tag{5}$$

This update process repeats until all T batches are processed, resulting in the final meta-memory $\mathcal{E}_T$.

Meta-Memory Update via Self-Reflection. We next detail how MetaMem performs self-reflection over the current meta-memory state to derive optimization actions, corresponding to Eq. 4.

For each instance $(q, a, \mathcal{M})$ in the batch $\tilde{\mathcal{D}}_t$ at optimization step t, MetaMem first prompts the LLM to randomly sample a set of responses $\mathcal{T} = \{\tau_1, \ldots, \tau_k\}$ conditioned on the current meta-memory state $\mathcal{E}_t$:

$$\tau_i \sim \mathrm{LLM}(\mathrm{Instruct}_{\mathrm{Gen}}(q, \mathcal{M}, \mathcal{E}_t)). \tag{6}$$

We then employ a stronger model as a judge to evaluate each sampled response $\tau_i \in \mathcal{T}$ against the ground-truth answer a:

$$s_i = \mathsf{Judge}(q, a, \mathcal{M}, \tau_i), \tag{7}$$

where s_i is a binary indicator. Specifically, $s_i = 1$ if τ_i matches the ground truth a_i, and $s_i = 0$ otherwise, indicating that τ_i is inconsistent with a_i. Based on the evaluation score s_i, we prompt the LLM to conduct self-reflection on each response τ_i, producing a reflection output r_i:

$$r_i = \mathrm{LLM}(\mathrm{Instruct}_{\mathrm{Reflect}}(q, a, \mathcal{M}, s_i, \tau_i)), \tag{8}$$

where $\mathrm{Instruct}_{\mathrm{Reflect}}$ guides the LLM to analyze the sampled response τ_i. Each reflection r_i includes an assessment of the LLM-generated responses, explaining the reasoning and underlying causes of

the correctness or incorrectness of τ_i. We aggregate these reflections into a set $R = \{r_1, \ldots, r_k\}$, which captures various insights over the sampled responses. We treat the reflection set R as feedback signals and leverage the instruction $\text{Instruct}_{\text{Action}}$ to prompt the LLM to generate an edit action o for the current meta-memory $\mathcal{E}_t$ at step t:

$$o = \text{LLM}(\text{Instruct}_{\text{Action}}(q, a, R, \mathcal{E}_t)). \tag{9}$$

We aggregate the actions o generated from all instances (q, a, R) in $\tilde{\mathcal{D}}_t$ to form an action set O_t. Since actions proposed by different instances may conflict, we further apply a filtering instruction, $\text{Instruct}_{\text{Filter}}$, to prompt the LLM to resolve such conflicts and produce a consistent action set $\tilde{O}_t$ for meta-memory update:

$$\tilde{O}_t = \text{LLM}(\text{Instruct}_{\text{Filter}}(O_t, \mathcal{E}_t)). \tag{10}$$

4 Experimental Methodology

This section describes the datasets, evaluation metrics, and baseline methods. Then, we introduce the implementation details of our experiments.

Datasets. We evaluate MetaMem on LongMemEval (Wu et al., 2024), a benchmark tailored for assessing long-horizon human-LLM interactions within multi-task dialogue scenarios. We additionally use the ShareGPT (ShareAI Lab, 2023) dataset as an out-of-domain corpus for MetaMem construction for a generalization test. More details are provided in Appendix A.2. Additionally, since LongMemEval contains only 500 test samples, we employ 5-fold cross-validation for evaluation, where each fold allocates 400 samples for Meta-Memory construction and 100 samples for testing. More details are shown in Appendix A.5.

Evaluation Metrics. Following Fang et al. (2025), we adopt accuracy as the evaluation metric, which is defined as the proportion of correctly answered questions. We evaluate MetaMem on the LongMemEval benchmark across four dimensions. Single-Session tasks (Single User, Single Assistant, and Single Preference) assess intra-session retention. Multi-Session tasks evaluate information integration across disjoint sessions. Temporal Reasoning tasks test the modeling of chronological dependencies. Knowledge Update tasks evaluate the capability of models to adapt to newly introduced conflicting information.

Baselines. We compare MetaMem with several representative baseline methods. The Full Text model directly feeds the entire dialogue history into the model as context for question answering. We further include RAG (Lewis et al., 2020) and MapReduce (Zhou et al., 2025) as baselines; both aim to identify and aggregate query-relevant chunks to produce the final response. Moreover, we compare against several memory-based systems that maintain persistent storage to track factual updates over long-term interactions, including Mem0 (Chhikara et al., 2025), A-Mem (Xu et al., 2025), MemoryOS (Kang et al., 2025), and LightMem (Fang et al., 2025). Specifically, Mem0 and A-Mem manage long-term interactions by maintaining persistent vector databases that automatically update user profiles and factual records. MemoryOS formulates memory management as an operating system, performing hierarchical read and write operations to ensure memory consistency and coherence. LightMem serves as our primary baseline; it optimizes knowledge storage by incorporating a text compression module, topic-aware short-term memory, and long-term memory maintenance mechanisms. For a fair comparison, all methods employ the same backbone LLM and retriever in our experiments.

Implementation Details. We conduct experiments using Qwen3-30B-A3B-Instruct (Yang et al., 2025) and Llama3.1-70B-Instruct (Grattafiori et al., 2024) as backbone models, and employ Qwen3-235B-A22B (Yang et al., 2025) for response judgment. The prompt templates used in MetaMem are provided in Appendix A.8. We leverage LightMem (Fang et al., 2025) for factual memory ($\mathcal{M}$) construction, and utilize all-MiniLM-L6-v2 (Wang et al., 2020) as the embedding model for memory retrieval. Further implementation details are provided in Appendix A.3.

5 Experimental Results

In this section, we show the overall performance of MetaMem and conduct ablation studies to investigate the contribution of each component. We then analyze the evolution of meta-memory units during the self-evolution process. Finally, case studies are provided in Appendix A.6.

5.1 Overall Performance

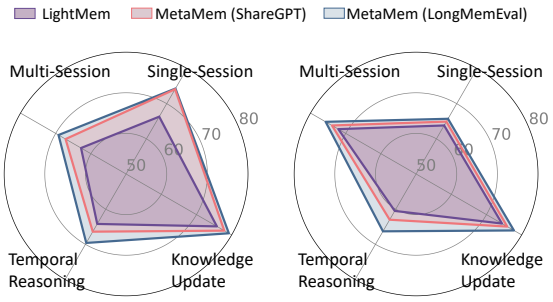
In this section, we first report the overall performance of MetaMem across two backbone models, followed by an assessment of its generalization capability in the out-of-domain setting.

Method	Single User	Single Assistant	Multi Session	Temporal Reasoning	Knowledge Update	Single Preference	Avg.
Qwen3-30B-A3B-Instruct							
Full Text	81.13	**88.37**	34.29	27.52	66.67	62.50	51.50
RAG	86.79	**88.37**	46.67	49.54	66.67	75.00	62.25
MapReduce (2025)	81.13	81.40	38.46	36.45	72.31	62.50	55.30
Mem0 (2025)	82.86	87.50	36.09	33.08	76.92	50.00	54.80
A-Mem (2025)	85.85	86.05	52.38	51.38	72.73	41.67	62.88
MemoryOS (2025)	69.81	79.07	37.14	28.44	59.09	29.17	46.75
LightMem (2025)	90.57	32.56	62.86	64.22	75.76	91.67	67.50
MetaMem	**90.70**	38.14	**69.24**	**69.60**	**79.18**	**94.16**	**71.90**
Llama3.1-70B-Instruct							
Full Text	58.49	72.09	21.90	22.02	57.58	33.33	38.75
RAG	84.91	**90.70**	48.57	47.71	62.12	58.33	60.50
MapReduce (2025)	86.79	83.72	15.24	26.61	71.21	62.50	47.25
Mem0 (2025)	81.43	89.29	34.15	35.88	77.18	38.33	54.62
A-Mem (2025)	84.29	88.64	56.72	57.45	73.56	32.50	62.88
MemoryOS (2025)	**90.57**	83.72	51.43	55.05	62.12	50.00	62.75
LightMem (2025)	90.36	22.36	75.31	52.52	76.92	**70.24**	66.17
MetaMem	90.28	25.64	**75.50**	**66.28**	**77.76**	66.12	**69.08**

Table 1: Overall Performance on LongMemEval (Wu et al., 2024) Benchmark. The best results are marked in **bold**, while the second-best results are underlined.

Table 1 presents the performance of MetaMem across the Qwen3-30B-A3B-Instruct and Llama3.1-70B-Instruct backbone models. Overall, MetaMem consistently achieves state-of-the-art performance, outperforming the strongest baseline by over 3.6%. Notably, MetaMem exhibits superior robustness on Multi-Session and Temporal Reasoning tasks, substantially outperforming our primary baseline LightMem (Fang et al., 2025). This result further confirms the effectiveness of the augmented meta-memory module. Compared to the Full Text baseline, chunking-based approaches such as RAG and MapReduce exhibit moderate improvements on Multi-Session and Temporal Reasoning tasks. However, their reliance on finding more query-related evidence leads to suboptimal performance in conducting static knowledge and continuous reasoning. Stateful memory systems, including Mem0, A-Mem, and MemoryOS, further improve performance on Single-User and Knowledge Update tasks by maintaining persistent knowledge across sessions. However, these methods update memory content in a largely static manner and are not tailored to the LLMs during the reasoning process, resulting in limited gains on complex long-horizon reasoning tasks. In contrast, MetaMem leverages a self-evolving meta-memory that is dynamically updated through feedback while handling different tasks, enabling LLMs to effectively utilize static knowledge stored in existing memories and produce more accurate responses.

Additionally, we train MetaMem separately on out-of-domain data (ShareGPT) and in-domain data (LongMemEval), resulting in two variant models, MetaMem (ShareGPT) and MetaMem (LongMemEval), to assess the cross-domain generalization capability of meta-memory construction across different datasets. As shown in Figure 3, both MetaMem (ShareGPT) and MetaMem (LongMemEval) consistently outperform the baseline LightMem model, regardless of whether the training data is in-domain or out-of-domain. These results indicate that the meta-memory learning process of MetaMem is not dependent on domain-specific data and exhibits strong generalization ability, effectively enhancing the LLM's capacity to leverage external memory. Moreover, MetaMem (LongMemEval), trained on in-domain data, achieves the best performance due to the closer alignment between training and evaluation,



(a) Preformance on Qwen3-30B-A3B-Instruct Model. (b) Preformance on Llama3.1-70B-Instruct Model.

Figure 3: Generalization Ability of Meta-Memory. Performance is reported under Qwen3-30B-A3B-Instruct and Llama3.1-70B-Instruct backbone models.

Method	Single User	Single Assistant	Multi Session	Temporal Reasoning	Knowledge Update	Single Preference	Avg.
Qwen3-30B-A3B-Instruct							
MetaMem	90.70	38.14	69.24	69.60	79.18	94.16	71.90
w/o Self-Reflection	90.57	34.35	66.24	68.20	78.19	91.67	70.02
w/o Evolution	90.57	32.56	64.71	66.67	76.92	91.67	68.33
Full Text	81.13	88.37	34.29	27.52	66.67	62.50	51.50
w/ MetaMem	86.79	88.37	38.10	25.69	69.70	62.50	53.25
RAG	86.79	88.37	46.67	49.54	66.67	75.00	62.25
w/ MetaMem	90.57	86.05	51.43	54.13	66.67	91.67	66.42
Llama3.1-70B-Instruct							
MetaMem	90.28	25.64	75.50	66.28	77.76	66.12	69.08
w/o Self-Reflection	90.28	25.64	75.42	64.68	77.34	69.45	68.33
w/o Evolution	90.11	23.88	75.42	63.15	77.34	69.45	66.42
Full Text	58.49	72.09	21.90	22.02	57.58	33.33	38.75
w/ MetaMem	59.84	72.09	23.15	23.88	59.21	36.67	40.42
RAG	84.91	90.70	48.57	47.71	62.12	58.33	60.50
w/ MetaMem	86.79	88.37	51.43	49.54	69.70	66.12	66.42

Table 2: Ablation Study of MetaMem.

highlighting the important role of a tailored meta-memory in effective memory utilization.

5.2 Ablation Study

To investigate the contribution of each component in MetaMem, we conduct an ablation study on the Qwen3-30B-A3B-Instruct and Llama3.1-70B-Instruct backbone models.

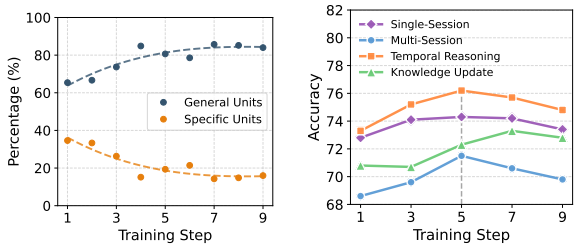
As shown in Table 2, we first evaluate two variants: MetaMem w/o Evolution, which disables the evolution of the meta-memory, and MetaMem w/o Self-Reflection, which removes the intermediate self-reflection step and directly generates optimization actions from the original responses. The evaluation results show that, compared with MetaMem w/o Evolution, MetaMem achieves more than 3% performance improvements, demonstrating the critical role of the iterative optimization mechanism in constructing a more tailored meta-memory to guide LLMs in effectively utilizing knowledge. These results suggest that meta-memory should be continuously updated based on feedback from dynamic environments, rather than remaining static. By learning across diverse tasks, the meta-memory is iteratively updated to capture generalizable knowledge and experience, enabling LLMs to handle different tasks more effectively instead of overfitting to task-specific factual knowledge. When removing the self-reflection step (MetaMem w/o Self-Reflection), the performance of MetaMem usually degrades, highlighting the effectiveness of the self-reflection module. This observation indicates that the self-reflection process complements the meta-memory update mechanism by functioning as a critic, producing high-quality insights through contrastive comparison between generated responses and ground-truth answers.

Furthermore, to assess the generalizability of MetaMem across different knowledge utilization systems, we apply MetaMem to two distinct methods, namely Full Text and RAG, resulting in two variants: Full Text (w/ MetaMem) and RAG (w/ MetaMem). Experimental results demonstrate that both variants achieve consistent performance improvements over vanilla memory models, showing their effectiveness and generalization capability. These results confirm that the effectiveness of MetaMem is decoupled from the underlying memory system, showing that MetaMem can robustly adapt to and enhance diverse memory systems without relying on system-specific characteristics.

5.3 Characteristics of Meta-Memory Evolution During Optimization

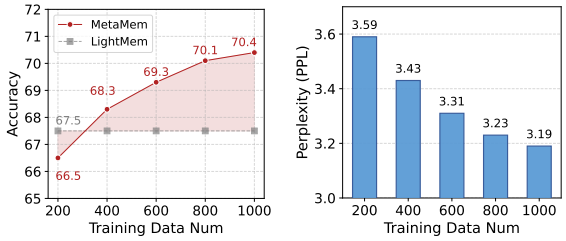
As shown in Figure 4, we analyze how memory usage experiences stored in the meta-memory evolve as training progresses, and examine the corresponding performance changes induced by the evolved meta-memory. This experiment adopts Qwen3-30B-A3B-Instruct as the backbone model.

As shown in Figure 4(a), we first investigate the evolution of meta-memory units throughout the training process. Specifically, we employ GLM-4.5 (Zeng et al., 2025) to classify memory usage experience units extracted from meta-memory checkpoints at each training step into two categories: Specific units, which capture task-specific memory usage experiences, and General units, which represent task-agnostic and broadly applicable experiences. Detailed classification instructions are



(a) Unit Categories of Meta-Memory During Training. (b) Accuracy Trends Across Different Sub-Tasks.

Figure 4: Performance of the Evolved Meta-Memory Across Different Training Steps.



(a) QA Accuracy. (b) Predictive Uncertainty.

Figure 5: Effectiveness of MetaMem Optimized with Different Data Scales. Figure 5(a) illustrates the model performance as the training dataset size increases. Figure 5(b) reports the perplexity of model outputs conditioned on the learned meta-memory.

provided in Appendix A.8. The results demonstrate that, as training proceeds, the proportion of General units increases steadily from approximately 65% to over 80%, and eventually stabilizes at this higher level. This observation indicates that the optimization process progressively shifts the meta-memory from encoding task-specific experiences toward consolidating more generalizable memory usage patterns, thereby enabling the memory system to transfer and apply learned experiences more effectively across diverse tasks.

Additionally, as shown in Figure 4(b), we report the task-wise accuracy of MetaMem under different stages of meta-memory evolution. The results show that during the early training stages, MetaMem exhibits consistent performance improvements across all tasks, suggesting that updating meta-memory using more general memory usage experiences effectively enhances the system's ability to leverage external memory. However, except for the Knowledge Update task, performance on most tasks begins to decline after peaking at step 5. This trend suggests that excessive training may cause the meta-memory to overfit specific knowledge update behaviors learned from later optimization, thereby degrading the performance of MetaMem.

5.4 Effectiveness of MetaMem with Human-LLM Interaction Scaling

To comprehensively evaluate the effectiveness of MetaMem under human-LLM interaction scaling, we report both task accuracy and predictive confidence in Figure 5. All models are implemented based on Qwen3-30B-A3B-Instruct and optimized using ShareGPT data for meta-memory training.

As illustrated in Figure 5(a), under extremely limited data regimes, MetaMem slightly underperforms the baseline. This initial degradation is likely due to the instability of the learned meta-memory during the early stage of optimization. Nevertheless, MetaMem quickly adapts and surpasses the baseline with only 400 training samples. As the data scale continues to increase, MetaMem maintains a stable upward performance trend, eventually widening the performance gap to 2.9% at 1,000 samples. These results indicate that MetaMem is highly data-efficient, demonstrating a strong ability to learn how to effectively facilitate knowledge utilization within the model as training data scales.

Furthermore, Figure 5(b) presents the evolution of perplexity. Overall, we observe a monotonic reduction in PPL from 3.59 to 3.19, suggesting that increased data exposure consistently reduces predictive uncertainty. Notably, the rate of improvement exhibits diminishing marginal returns: the PPL decreases substantially when scaling the data from 200 to 800 samples, but largely plateaus between 800 and 1,000 samples. This observation suggests that the core meta-memory patterns can be effectively acquired with a moderate amount of data, enabling LLMs to better leverage memorized knowledge and generate more confident responses.

6 Conclusion

This paper proposes MetaMem, a novel framework that equips LLM-based memory systems with a self-evolving Meta-Memory to effectively teach LLMs to learn to use memorized knowledge. Extensive experimental results demonstrate that MetaMem achieves state-of-the-art performance, while exhibiting strong cross-domain generalization and scalability, thereby benefiting long-horizon human-LLM interactions.

Limitations

Although our approach demonstrates convincing performance in guiding LLMs to utilize knowledge effectively, it has some limitations. First, our approach relies on an LLM-based judgment model to evaluate complex human-LLM interactions during meta-memory construction. Since real-world human-LLM interactions are highly nuanced and laden with diverse, implicit signals, establishing an accurate evaluation mechanism remains a persistent challenge. Second, although meta-memory trained on out-of-domain data demonstrates strong generalization across diverse scenarios, our experiments show that it remains inferior to in-domain training. In practice, however, high-quality data for specific professional scenarios is often limited, making it difficult to fully optimize a scenario-specific meta-memory. Consequently, we resort to a general-purpose meta-memory trained on large-scale heterogeneous data as a practical compromise, which prioritizes broad applicability.

Ethical Considerations

This work does not raise significant ethical concerns. The experiments rely on the ShareGPT dataset, which is publicly available and has undergone strict anonymization and de-identification processes. We ensure that the data used in this study does not contain any personally identifiable information (PII) or sensitive private content. Consequently, our research involves no privacy risks regarding real-world users. Furthermore, our proposed method focuses on optimizing memory mechanisms and does not introduce additional risks regarding the generation of harmful or biased content. All experiments are conducted in compliance with standard ethical guidelines for academic research.

References

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *ArXiv preprint*, abs/2504.19413.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, and 1 others. 2025. Lightmem: Lightweight and efficient memory-augmented generation. *ArXiv preprint*, abs/2510.18866.

John H Flavell. 1979. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American psychologist*, 34(10):906.

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

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021. Efficient nearest neighbor language models. In *Proceedings of EMNLP*, pages 5703–5714, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of NeurIPS*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024a. Deepseek-v3 technical report. *ArXiv preprint*, abs/2412.19437.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, and 1 others. 2025a. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Zhenghao Liu, Pengcheng Huang, Zhipeng Xu, Xinze Li, Shuliang Liu, Chunyi Peng, Haidong Xin, Yukun Yan, Shuo Wang, Xu Han, and 1 others. 2025b. Knowledge intensive agents. *Available at SSRN 5459034*.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. Memgpt: Towards llms as operating systems. *ArXiv preprint*, abs/2310.08560.

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 UIST*, pages 1–22.

ShareAI Lab. 2023. Sharegpt-chinese-english-90k: A bilingual chinese-english human-machine dialogue dataset. https://huggingface.co/datasets/shareAI/ShareGPT-Chinese-English-90k. Hugging Face dataset repository.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *ArXiv preprint*, abs/2507.20534.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of ACL*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings of NeurIPS*.

Xindi Wang, Mahsa Salmani, Parsa Omidi, Xiangyu Ren, Mehdi Rezagholizadeh, and Armaghan Eshaghi. 2024. Beyond the limits: A survey of techniques to extend the context length in large language models. In *Proceedings of IJCAI*, pages 8299–8307. ijcai.org.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2024. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *ArXiv preprint*, abs/2410.10813.

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

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

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, and 1 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *ArXiv preprint*, abs/2508.06471.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. 2025. G-memory: Tracing hierarchical memory for multi-agent systems. *ArXiv preprint*, abs/2506.07398.

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

Zihan Zhou, Chong Li, Xinyi Chen, Shuo Wang, Yu Chao, Zhili Li, Haoyu Wang, Qi Shi, Zhixing Tan, Xu Han, and 1 others. 2025. Llm× mapreduce: Simplified long-sequence processing using large language models. In *Proceedings of ACL*, pages 27664–27678.

Task	Data Num.
Single-Session User	70
Single-Session Assistant	56
Multi-Session	133
Temporal Reasoning	133
Knowledge Update	78
Single-Session Preference	30

Table 3: Statistics of LongMemEval Benchmark.

A Appendix

A.1 License

All datasets used in this work are released under licenses that permit academic research and non-commercial use, including LongMemEval (MIT License) and ShareGPT (Apache License 2.0).

A.2 Additional Dataset Details.

We employ both the LongMemEval (Wu et al., 2024) and ShareGPT (ShareAI Lab, 2023) datasets for the construction and evaluation of MetaMem. For LongMemEval, we adhere to the evaluation protocol established by Fang et al. (2025), with detailed dataset statistics presented in Table 3. For ShareGPT, which serves as an out-of-domain corpus for MetaMem construction, we filter conversations that exceed 8 turns and randomly sample 1,000 instances with a random sample seed of 42. During inference, the LLM is provided with the first $n-1$ turns of a conversation and tasked with predicting the final user question.

A.3 Additional Experimental Details

For all the baselines, we utilize the same generation settings during memory construction. Specifically, we employ all-MiniLM-L6-v2 (Wang et al., 2020) as the embedding model and set the retrieval top-k to 20. For the RAG method, each dialogue turn is treated as a distinct chunk, and we concatenate the timestamp information to each chunk before embedding. During inference, we utilize the SGLang[1] framework to deploy the LLM server for high-performance inference. We standardize the LLM sampling parameters with a temperature of 0.0, a top-p of 0.8, and a maximum token limit of 2000. During meta-memory construction, we maintain the sampling parameters at a fixed temperature of 0.7, a top-p of 0.95, and a maximum output length of 4000 tokens, performing 5 samples per instance. The model is trained for 5 epochs with a batch size of 50.

[1] https://github.com/sgl-project/sglang

Method	Output Token	Inference Time (s)
Qwen3-30B-A3B-Instruct		
Full Text	252.72	10.95
w/ MetaMem	177.01	8.52
RAG	297.96	18.60
w/ MetaMem	339.53	21.05
LightMem	175.52	17.59
w/ MetaMem	184.72	19.87
Llama3.1-70B-Instruct		
Full Text	572.38	87.45
w/ MetaMem	497.31	74.83
RAG	80.89	10.77
w/ MetaMem	96.73	13.05
LightMem	84.80	11.23
w/ MetaMem	111.82	14.86

Table 4: Inference Efficiency. We evaluate the average generated tokens and inference time of MetaMem against several baselines. All experiments are conducted on the same device, with memory construction steps executed offline.

A.4 Analysis of Inference Efficiency.

Table 4 reports the inference efficiency of MetaMem across Qwen3-30B-A3B-Instruct and Llama3.1-70B-Instruct backbone models. We implement MetaMem across three distinct baselines-Full Text, RAG, and LightMem-to explicitly evaluate the inference latency overhead introduced by our approach. In the Full Text setting, MetaMem notably accelerates inference by curbing verbosity and hallucinations, resulting in reduced token counts and latency. For the chunking-based method (RAG) and the stateful memory system (LightMem), MetaMem incurs negligible overhead due to the encoding of additional meta-directives. These results confirm that MetaMem enhances response quality with minimal computational cost.

A.5 Details of 5-Fold Cross-Validation

We utilized random sampling to partition the dataset into five distinct subsets. To ensure reproducibility, we set the random sample seed to 42. For each fold, a subset of 100 samples is reserved as the independent test set, while the remaining data is partitioned into 350 samples for meta-memory training and 50 samples for validation. Table 5 shows the detailed results of our 5-fold cross-validation. We reported the final results by calculating the average performance across these five folds.

A.6 Case Study

In Table 6, we present a qualitative case study to illustrate how the learned meta-memory effectively

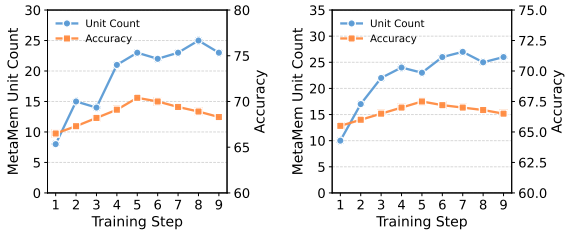
Fold Index	Method	Single User	Single Assistant	Multi Session	Temporal Reasoning	Knowledge Update	Single Preference	Avg.
Qwen3-30B-A3B-Instruct								
Fold 1	Baseline	100.0	37.5	61.1	51.5	82.6	100.0	67.7
	MetaMem	91.7	37.5	55.6	60.6	91.3	100.0	70.7
Fold 2	Baseline	100.0	45.5	72.0	58.3	88.9	71.4	68.7
	MetaMem	90.9	54.5	88.0	66.7	88.9	100.0	77.8
Fold 3	Baseline	100.0	33.3	65.6	75.0	72.2	83.3	69.7
	MetaMem	90.9	33.3	78.1	80.0	83.3	83.3	75.8
Fold 4	Baseline	93.8	50.0	68.0	36.0	53.8	75.0	60.6
	MetaMem	100.0	50.0	60.0	76.0	53.8	87.5	70.7
Fold 5	Baseline	75.0	7.7	64.5	70.6	71.4	100.0	62.6
	MetaMem	80.0	15.4	64.5	64.7	78.6	100.0	64.6
Llama3.1-70B-Instruct								
Fold 1	Baseline	100.0	25.0	61.1	54.5	78.3	100.0	66.7
	MetaMem	91.7	25.0	55.6	54.5	78.3	80.0	63.6
Fold 2	Baseline	90.9	27.3	84.0	58.3	88.9	42.9	66.7
	MetaMem	100.0	45.5	80.0	69.4	77.8	71.4	73.7
Fold 3	Baseline	90.9	16.7	84.4	55.0	77.8	83.3	69.7
	MetaMem	90.9	25.0	81.3	75.0	77.8	66.7	72.7
Fold 4	Baseline	100.0	33.3	76.0	36.0	84.6	75.0	65.7
	MetaMem	93.8	25.0	80.0	56.0	69.2	62.5	66.7
Fold 5	Baseline	70.0	7.7	71.0	58.8	85.7	50.0	61.6
	MetaMem	75.0	7.7	80.6	76.5	85.7	50.0	68.7

Table 5: 5-Fold Cross-Validation.

guides LLM to utilize knowledge from both aggregated facts and fragmented numerical evidence during long-horizon interaction sessions.

The user queries the total distance driven across four road trips. The retrieved memory fragments contain a mixture of numerical signals at different granularities, such as explicit summaries (*e.g.*, "1,800 miles across three trips") and constituent details (*e.g.*, "300 miles on the first day"). The baseline model (LightMem (Fang et al., 2025)) fails to comprehend the inclusion relationship between them. It treats the partial distance (300 miles) as an independent event rather than a subset of the total trip, leading to erroneous double-counting. Furthermore, it fails to filter out irrelevant planning-related estimates, resulting in a chaotic and incorrect summation.

In contrast, MetaMem successfully leverages a learned meta-memory unit that instructs the model to prioritize verified cumulative statistics over fragmented partial data. Guided by this experience, the model correctly identifies that the "300-mile" segment is inherently encompassed within the "1,200-mile Yellowstone trip" summary. Consequently, MetaMem filters out the redundant partial values and correctly composes the two high-level aggregates (1,800 miles + 1,200 miles), yielding the precise total. This case demonstrates that the learned meta-memory equips the LLM with the ability to perform hierarchical reasoning, effectively utilizing retrieved knowledge by pruning redundant or subordinate information.



(a) Qwen3-30B-A3B-Instruct. (b) Llama3.1-70B-Instruct.

Figure 6: Evolution of Meta-Memory Unit Count and Task Performance.

A.7 Evolution of Meta-Memory Unit Count and Task Performance

We further investigate the correlation between the count of the learned meta-memory units and task performance during the training process. Figure 6 illustrates the evolution of unit counts and accuracy on the Qwen3-30B-A3B-Instruct and Llama3.1-70B-Instruct backbones.

As observed in both model backbones, the number of meta-memory units exhibits a rapid growth phase in the early stages. This accumulation of units correlates positively with performance improvements, indicating that the model is effectively acquiring necessary reasoning directives. However, a turning point appears around step 5. While

the unit count continues to rise or stabilize at a high level (approximately 25-27 units), the accuracy peaks at step 5 and subsequently exhibits a downward trend. This suggests that beyond this optimal point, the continued accumulation of rules may introduce redundancy or overly specific constraints, leading to overfitting and a slight degradation in general reasoning performance.

A.8 Instruction Prompts

This section presents the detailed prompt templates utilized throughout our experiments. Specifically, Figures 7-11 illustrate the prompts used for the LLM judge evaluation. Figure 12 displays the prompt for answer generation ($\text{Instruct}_{\text{Gen}}$). Figures 14-16 detail the prompts for meta-memory construction ($\text{Instruct}_{\text{Reflect}}$, $\text{Instruct}_{\text{Action}}$, and $\text{Instruct}_{\text{Filter}}$). Finally, Figure 13 presents the prompt used for the classification of meta-memory units.

Case: Cumulative Numerical Reasoning: Road Trip Distance

Question: What is the total distance I have driven across my four road trips? **Ground Truth Answer:** 3000 miles

Related Memories:

2023/05/22 (Mon) 07:27 ... charity run distances ... 5K / 10K / marathon ...

2023/05/23 (Tue) 16:14 ... I've covered a total of 1,800 miles on my recent three road trips, including a solo trip to Durango, a weekend trip to Breckenridge, and a family trip to Santa Fe ...

2023/05/23 (Tue) 16:14 ... stopping at Maroon Lake ... how long does it take ... worth the detour ...

2023/05/23 (Tue) 16:14 ... planning a road trip from Denver to Aspen ... scenic stops ...

2023/05/26 (Fri) 14:27 ... another road trip ... just got back from an amazing 4-day trip to Yellowstone National Park ... where we covered a total of 1,200 miles.

2023/05/26 (Fri) 14:27 ... thinking about our Yellowstone trip last month ... we drove around 300 miles on the first day to reach Jackson, Wyoming ...

2023/05/26 (Fri) 14:27 ... planning road trip from Denver to Mount Rushmore ... approximately 570 miles ...

Response of Baseline:

Based on the conversation records, you have driven a total of 2230 miles across four road trips. Below is a breakdown of the distances according to your own descriptions:

1. Yosemite National Park trip: 1,200 miles over four days.

2. Denver to Aspen: 160 miles.

3. Denver to Mount Rushmore: 570 miles.

4. California to Florida trip: only the first day (300 miles to Jackson, Wyoming) was mentioned.

Summation: 1200 + 160 + 570 + 300 = 2230 miles.

The initial statement that three trips totaled 1800 miles introduces ambiguity, but since the California–Florida trip lacks a full distance, only explicitly stated values are used. Therefore, the computed total is 2230 miles.

Response of MetaMem:

The total distance is obtained by summing explicitly stated trips:

(1) Three trips (Durango, Breckenridge, Santa Fe) explicitly stated to total 1800 miles.

(2) A separate four-day Yellowstone National Park trip with 1200 miles.

The mention of driving "300 miles on the first day" is a partial segment already included in the 1200-mile Yellowstone total, so it is not added separately. Similarly, the Mount Rushmore trip is a future plan, not a completed trip.

Thus, the total distance is 1800 + 1200 = 3000 miles.

Used MetaMem Unit:

(1) When performing numerical aggregation, prioritize explicit high-level summaries or total statistics provided by the user over fragmented partial details (e.g., daily breakdowns or single segments).

(2) Identify inclusion relationships to avoid double-counting subsets that are already encompassed within a reported total.

Table 6: Case Study Comparing LightMem and MetaMem. Correctly utilized information is highlighted in green, while misleading or erroneous elements are highlighted in orange.

Judge Instruction for Single-User, Single-Assistant and Multi-Session Task

I will give you a question, a correct answer, and a response from a model. Please answer yes if the response contains the correct answer. Otherwise, answer no. If the response is equivalent to the correct answer or contains all the intermediate steps to get the correct answer, you should also answer yes. If the response only contains a subset of the information required by the answer, answer no.

Question: {question}

Correct Answer: {ground_truth}

Model Response: {pred_answer}

Is the model response correct? Answer yes or no only.

Figure 7: Judge Instruction for Single-User, Single-Assistant and Multi-Session Task.

Judge Instruction for Temporal-Reasoning Task

I will give you a question, a correct answer, and a response from a model. Please answer yes if the response contains the correct answer. Otherwise, answer no. If the response is equivalent to the correct answer or contains all the intermediate steps to get the correct answer, you should also answer yes. If the response only contains a subset of the information required by the answer, answer no. In addition, do not penalize off-by-one errors for the number of days. If the question asks for the number of days/weeks/months, etc., and the model makes off-by-one errors (e.g., predicting 19 days when the answer is 18), the model's response is still correct.

Question: {question}

Correct Answer: {ground_truth}

Model Response: {pred_answer}

Is the model response correct? Answer yes or no only.

Figure 8: Judge Instruction for Temporal-Reasoning Task.

Judge Instruction for Knowledge-Update Task

I will give you a question, a correct answer, and a response from a model. Please answer yes if the response contains the correct answer. Otherwise, answer no. If the response contains some previous information along with an updated answer, the response should be considered as correct as long as the updated answer is the required answer.

Question: {question}

Correct Answer: {ground_truth}

Model Response: {pred_answer}

Is the model response correct? Answer yes or no only.

Figure 9: Judge Instruction for Knowledge-Update Task.

Judge Instruction for Single-Session-Preference Task

I will give you a question, a rubric for desired personalized response, and a response from a model. Please answer yes if the response satisfies the desired response. Otherwise, answer no. The model does not need to reflect all the points in the rubric. The response is correct as long as it recalls and utilizes the user's personal information correctly.

Question: {question}

Rubric: {ground_truth}

Model Response: {pred_answer}

Is the model response correct? Answer yes or no only.

Figure 10: Judge Instruction for Single-Session-Preference Task.

Judge Instruction for Unanswerable Task

I will give you an unanswerable question, an explanation, and a response from a model. Please answer yes if the model correctly identifies the question as unanswerable. The model could say that the information is incomplete, or some other information is given but the asked information is not.

Question: {question}

Explanation: {ground_truth}

Model Response: {pred_answer}

Does the model correctly identify the question as unanswerable? Answer yes or no only.

Figure 11: Judge Instruction for Unanswerable Task.

Inference Instruction of MetaMem

You are an intelligent assistant with access to a memory system. Your task is to answer the user's question by effectively utilizing the retrieved memory fragments.

Meta-Memory Guidelines:
The following are learned strategies that teach you how to effectively utilize memory fragments. Apply these guidelines when processing the retrieved memories:\n{meta_memories}

User Question:\n{question}

Retrieved Memory Fragments:\n{memories}

Instructions:
1. First, review the meta-memory guidelines to understand how to approach memory utilization
2. Analyze the retrieved memory fragments and identify relevant information
3. Apply the meta-memory strategies to synthesize information from memories
4. Formulate your answer based on the synthesized knowledge
5. If memories are insufficient or conflicting, handle according to the guidelines

Think step by step about how to utilize the memories, then provide your final answer.

Figure 12: Inference Instruction of MetaMem.

Instruction for Classifying the Meta Memory Units

You are a data classifier analyzing guidelines from an meta memory.
Classify the following guideline based on its Scope of Applicability:

1. UNIVERSAL_LOGIC (General):
- The guideline describes a general reasoning logic that can be applied to ALMOST ANY topic (e.g., temporal reasoning, handling negation, counting entities, resolving pronouns, detecting causal root causes).
- It does NOT rely on specific domain knowledge (like how camping works, or specific video game genres).
- It is "content-agnostic".

2. DOMAIN_HEURISTIC (Specific):
- The guideline is tailored to a specific topic, activity, or domain (e.g., camping, pets, video games, travel, creative writing, health).
- It mentions specific keywords or logic that only makes sense within that specific context.
- Example: "When inferring camping...", "If the user mentions a game...", "For pet care...".

Guideline: {text}

Reply ONLY with the word "GENERAL" or "SPECIFIC". Do not add any explanation.

Figure 13: Instruction for Classifying the Meta Memory Units .

Instruction for Response Reflection

You are analyzing a model's response to a memory-based question-answering task. Your goal is to summarize the reasoning trajectory and identify the key factors that led to a correct or incorrect answer.

Task Context:
<question>{question}</question>\n\n
<retrieved_memories>{memories}</retrieved_memories>\n\n
<ground_truth_answer>{answer}</ground_truth_answer>\n\n
<model_response>{response}</model_response>\n\n
<evaluation>Correct: {grade}</evaluation>\n\n

Analysis Instructions:
1. Memory Utilization Analysis:
- Which memories did the model use? Which did it ignore?
- Was the memory selection appropriate for answering the question?
- Did the model correctly interpret the information in the memories?
2. Reasoning Process:
- What reasoning steps did the model take?
- Were there any logical errors or correct inferences?
- How did the model synthesize information from multiple memories?
3. Key Decision Points:
- What were the critical decisions that led to success or failure?
- If correct: What memory utilization strategy worked well?
- If incorrect: Where did the reasoning go wrong? What information was missed or misinterpreted?

Provide a concise summary focusing on the above points:

Figure 14: Instruction for Response Reflection.

Instruction for Generating Meta Memory Update Actions

You are a meta-learning specialist. Your task is to analyze multiple attempts at answering the same question and derive generalizable meta-memory principles. Meta-memories are high-level strategies that teach a model "learning to learn". Specifically, how to effectively utilize retrieved memory fragments to answer questions.

Question:\n{question}\n\n
Retrieved Memories:\n{memories}\n\n
Ground Truth Answer:\n{answer}\n\n
Summaries of Multiple Attempts:\n{summaries}\n\n
Current Meta-Memory Knowledge Base:\n{meta_memories}\n\n

Your Task:
1. Cross-Attempt Analysis:
- Compare the successful vs unsuccessful attempts
- Identify patterns: What strategies led to correct answers?
- Identify anti-patterns: What mistakes led to incorrect answers?
2. Derive Meta-Memory Principles:
Based on your analysis, propose updates to the meta-memory knowledge base.
Each meta-memory should be:
- A generalizable strategy about HOW to utilize memories (not domain-specific facts)
- Actionable guidance that can be applied to future questions
- Concise (one sentence, max 30 words)
Examples of good meta-memories:
- "When memories contain temporal information, prioritize the most recent data unless the question asks about history."
- "Cross-validate facts that appear in multiple memories; single-source claims require more caution."
- "If memories seem contradictory, check if they refer to different time periods or contexts."
3. Propose Operations:
- add: Add a new meta-memory (when you discover a new principle not covered by existing ones)
- update: Update an existing meta-memory by ID (when you can improve or refine an existing principle)
- delete: Delete an existing meta-memory by ID (when a principle is wrong or redundant)

Output Format:
First, provide your reasoning. Then output a JSON array:
```json
[
  {{"operation": "add", "content": "Your new meta-memory principle"}},
  {{"operation": "update", "id": "M0", "content": "Updated meta-memory principle"}},
  {{"operation": "delete", "id": "M1"}}
]
```
Note: \nQuality over quantity - only propose operations that genuinely improve the meta-memory knowledge base.

Figure 15: Instruction for Generating Meta Memory Update Actions.

Instruction for Filtering Meta Memory Update Actions

You are a meta-learning specialist responsible for consolidating meta-memory updates from multiple learning samples. A batch of questions has been analyzed, and each analysis proposed some operations to update the meta-memory knowledge base.
Your task is to consolidate these proposals, resolve conflicts, and produce the final update plan.

Current Meta-Memory Knowledge Base:\n{existing_meta_memories}\n\n
Proposed Operations from This Batch:\n{proposed_updates}\n\n

Consolidation Guidelines:
1. Merge Similar Proposals:
- If multiple proposals suggest similar principles, merge them into a single, more general formulation
- Prefer broader applicability over narrow specificity
2. Resolve Conflicts:
- If proposals contradict each other, analyze which is more generally valid
- Consider how many different questions support each proposal
- When in doubt, prefer the more cautious/conservative principle
3. Avoid Redundancy:
- Do not add a new meta-memory if it overlaps significantly with an existing one
- Instead, update the existing one to incorporate the new insight
4. Quality Control:
- Each final meta-memory must be about memory utilization strategy
- Each must be generalizable (not specific to one question type)
- Each must be actionable and concise (max 30 words)
5. Maintain Stability:
- Don't delete meta-memories unless they are clearly wrong or completely redundant
- Prefer updating over deleting + adding

Output Format:
First, analyze the proposed operations and explain your consolidation decisions.
Then output the final operations to apply:
```json
[
  {{"operation": "add", "content": "Consolidated new meta-memory"}},
  {{"operation": "update", "id": "M0", "content": "Consolidated updated meta-memory"}},
  {{"operation": "delete", "id": "M2"}}
]
```
Note: The number of final operations should typically be less than the number of proposals due to consolidation.

Figure 16: Instruction for Filtering Meta Memory Update Actions.