SCM: Enhancing Large Language Model with Self-Controlled Memory Framework

Bing Wang[1], Xinnian Liang[1], Jian Yang[1](✉), Hui Huang[3], Zhenhe Wu[1], ShuangZhi Wu[2], Zejun Ma[2], and Zhoujun Li[1]

[1] Beihang University
{bingwang,xnliang,jiaya,wuzhenhe,lizj}@buaa.edu.cn
[2] Bytedance
{wufurui,mazejun}@bytedance.com
[3] Harbin Institute of Technology
huanghui@stu.hit.edu.cn

Abstract. Large Language Models (LLMs) are constrained by their lack of a long-term memory mechanism, which hinders their ability to maintain context over extended periods and leads to the loss of crucial historical information. To address this limitation, in this paper, we propose the Self-Controlled Memory (SCM) framework to enhance the ability of LLMs to maintain long-term memory and recall relevant information. Our SCM framework comprises three key components: *an LLM-based agent* serving as the backbone of the framework, *a memory stream* storing agent memories, and *a memory controller* updating memories and determining when and how to use the memories from the memory stream. Furthermore, we annotate a dataset, MemoEval, to assess the efficiency of SCM in utilizing memories and processing lengthy inputs. The MemoEval dataset covers three tasks: long-term dialogues, book summarization, and meeting summarization. Experimental results reveal that our SCM framework significantly increases overall accuracy by about 40% compared to vanilla ChatGPT on the long-term dialogue task[1].

Keywords: Large language model · self-controlled memory · memory retrieval.

1 Introduction

Recently, Large Language Models (LLMs) have attracted significant attention due to their remarkable performance in various tasks [6, 32]. Instruction-tuning [25] helps LLMs comprehend natural language task descriptions, while reinforcement learning with human feedback [22] aligns the generated text with human preferences. LLMs offer numerous advantages, but their utility is hindered by two main factors: the absence of long-term memory and the limited context window length [15]. Although many LLMs [16, 4] are capable of processing long inputs, they still struggle to capture crucial contextual information in extremely long

[1] code: https://github.com/wbbeyourself/SCM4LLMs

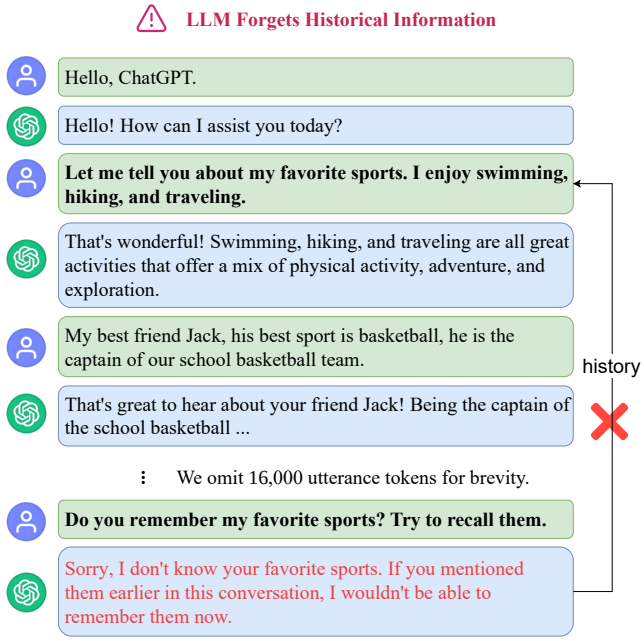

Fig. 1. An example demonstrating how LLM forgets historical information details.

texts [13]. As illustrated in Figure 1, even ChatGPT can forget crucial contextual information from the preceding text due to the accumulation of historical noise.

To address this limitation, we propose the **S**elf-**C**ontrolled **M**emory (**SCM**) framework, enhancing the ability of LLMs to maintain long-term memory and retrieve pertinent information as needed. Our SCM framework is built upon three critical elements: *an LLM-based agent* forming the core, *a memory stream* for chronicling the agent's historical data, and *a memory controller* tasked with the curation and strategic deployment of the memory. Within this construct, the input text is dissected into discrete segments and subsequently fed into the LM agent as individual observations. The LM agent processes each segment with a dual-memory approach: leveraging a persistent long-term memory (activation memory), which archives contextual history, juxtaposed against a transient short-term memory (flash memory) designed to resonate with immediate precedents. Concurrently, the memory controller exercises prudence in memory retrieval, selectively sourcing pivotal information and conscientiously sidestepping extraneous data interference.

Furthermore, we annotate the MemoEval dataset to evaluate the effectiveness of SCM to utilize memories and process long inputs. To ensure the authenticity and relevance of our constructed dataset, the MemoEval dataset is developed by collecting data from real user scenarios. The MemoEval dataset covers three

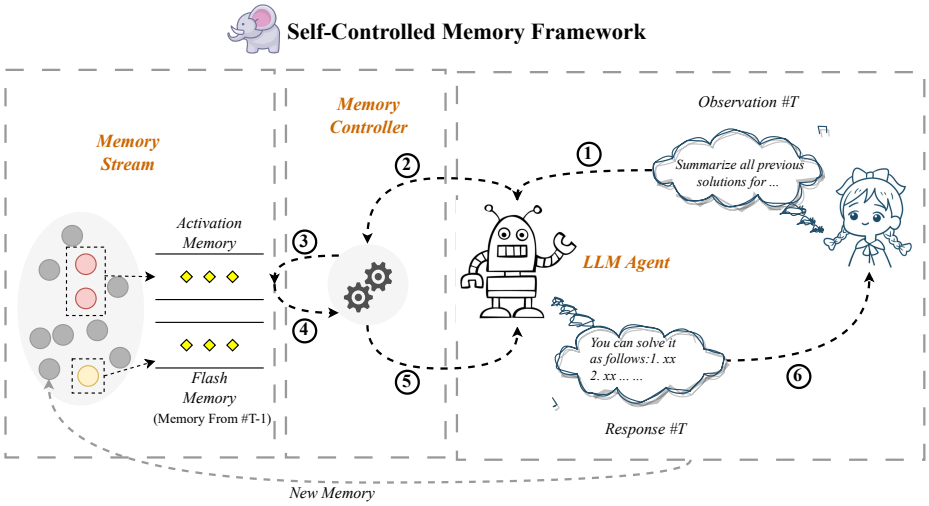

Fig. 2. The workflow of our proposed Self-Controlled Memory(SCM) framework, where numbers 1-6 represent the six explicit steps of one iteration with new observation #T. These steps are (1) Input Acquisition, (2) Memory Activation, (3) Memory Retrieval, (4) Memory Reorganization, (5) Input Fusion, and (6) Response Generation. Under this framework, the input text is divided into segments and then provided to the LLM as observations (inputs). Each segment is processed by the LLM using two types of memory: a long-term memory (activation memory) that retains historical information and a short-term memory (flash memory) that captures real-time memory information from the preceding segment.

tasks: long-term dialogues, book summarization, and meeting summarization. Extensive experimental results on the MemoEval dataset demonstrate that integration of the SCM framework with `text-davinci-003` significantly enhances performance, outperforming the vanilla ChatGPT by 40% in overall accuracy in long-term dialogue tasks. Moreover, when applied to summarization tasks, our SCM-based methods demonstrate markedly superior performance in generating summaries, with notable improvements in coherence and coverage compared to the baseline model.

In summary, our main contributions are as follows.

- We propose the self-controlled memory (SCM) framework to enhance the ability of LLMs to maintain long-term memory and recall relevant information.
- We contribute the MemoEval dataset to evaluate the effectiveness of SCM in three tasks: long-term dialogues, book summarization, and meeting summarization.
- Experimental results reveal that our SCM framework significantly increases overall accuracy by about 40% compared to vanilla ChatGPT on the long-term dialogue task.

2 Self-Controlled Memory

In this section, we provide a detailed description of our proposed self-controlled memory (SCM) framework, as illustrated in Figure 2. Firstly, the SCM workflow will be briefly introduced in Section 2.1. Subsequently, the three key components of SCM will be presented: (1) an LLM-based agent (Sec. 2.2) serving as the backbone of the framework, (2) a memory stream (Sec. 2.3) storing agent memories, and (3) a memory controller (Sec. 2.4) updating memories and determining when and how to use the memories from the memory stream.

2.1 Workflow of SCM

As illustrated in Figure 2, the SCM workflow consists of six explicit steps. Initially, the agent acquires observation at the turn T. Following this, the memory activation process begins, where the memory controller determines if it is necessary to activate memory based on the current observation. The memory retrieval is then initiated, using the observation as a query to retrieve the top K-ranked memories. The fourth step involves memory reorganization, in which the controller decides whether to use the original or summarized memory directly. Subsequently, the framework combines the retrieved memories in a predefined format, providing background information for response generation. The fifth step, input fusion, involves the predefined prompt that fuses the restructured memory with the present observation, serving as the model's input. Lastly, the LLM-based agent generates a response based on the result of the previous step, incorporating the current interaction, including the observation and the response, into the memory stream.

2.2 LLM-based Agent

LLM-based agents are featured in their self-evolving capability, which is the basis for solving real-world problems that need long-term and complex agent-environment interactions. The key component to support agent-environment interactions is the memory of the agents. The LLM-based agent serves as the core component of our SCM framework by generating coherent and accurate responses based on well-designed instructions. In this work, we adopt two powerful LLMs, *text-davinci-003* and *gpt-3.5-turbo*, as agents in our SCM framework, respectively.

2.3 Memory Stream

The memory stream stores all historical memory items and can easily achieve high-speed access through cache storage technologies such as Redis or vector databases such as Pinecone [2]. When memory retrieval is necessary, the memory stream retrieves and returns two kinds of items: **Activation Memory**, which

[2] `https://www.pinecone.io/`

stores related historical memories, and **Flash Memory**, which stores interaction memories of the previous turn $T-1$. Specifically, each memory item consists of (1) an interaction index, (2) an observation, (3) a system response, (4) a memory summarization (refer to the next paragraph for elaboration), and (5) an interaction embedding that illustrates the current interaction semantics.

Memory Summarization The motivation for using memory summarization is that the context window length of LLM is limited. Within the constraints of the length of the model context window, it is necessary to compress memory information effectively. Memory summarization plays a vital role in processing lengthy inputs, where a single interaction or dialogue turn can consist of more than 3,000 tokens. Obtaining the key information of individual turns through turn summarization is a non-trivial task when attempting to integrate multi-turn information within a limited contextual window.

Memory Reorganization In our study, we employ an empirical approach of concatenating the observation summary and system response summary (i.e., the memory summarization result of each item) to derive semantic representations for individual items. This concatenation is necessary due to the potential significant variation in length between the observation and system response within the memory stream.

2.4 Memory Controller

This section focuses on the central component: the memory controller, and its workflow is illustrated in Figure 3. The primary objective behind the design of the memory controller is to introduce the minimum necessary information to avoid excessive noise that may disrupt the model's performance.

Specifically, this can be divided into three scenarios for discussion. First, not all observations, also referred to as user input or instruction, require access to historical memory. For example, the user instruction "Tell me a joke" does not require retrieving the user's historical memory. However, certain user input, such as "Do you remember the conclusion we made last week on the fitness diets" requires the retrieval of memories. Secondly, the amount of memory can be enormous, ranging from hundreds to thousands or even tens of thousands. A controller is needed to retrieve and filter the memory. Third, given the model's limited input length, the controller must determine whether to employ the full content of the memory or a summary of it. The original full text can be excessively long and may exceed the model's maximum length capacity. Subsequent subsections present the detailed workflow of the memory controller, which considers each of the aforementioned scenarios.

Memory Controller Workflow As illustrated in Figure 3, the memory controller determines when to retrieve memories and how to utilize the retrieved memories in response to a novel observation.

The controller is also a language model, which controls the entire process by self-asking two questions:

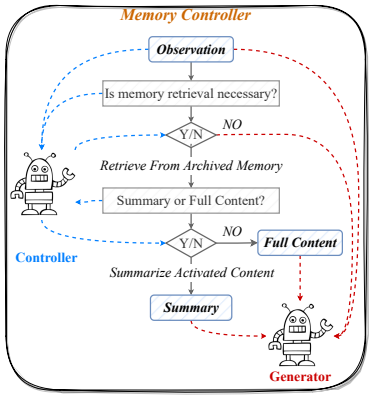

Fig. 3. Workflow of the Memory Controller.

1. Is it necessary to activate the memories given the current user input?
2. Can the current user input be answered correctly using only the memory summary?

Activate Memories To address the first question, we have devised a controller-specific instruction to determine whether to activate memory. If the model responds with "yes (A)", the relevant memories will be activated to provide an answer to the current question. During the retrieval process of memories, we use current observation as a query and assess the rank score of each memory based on relevance. The relevance score of each memory is computed by calculating the cosine similarity between the current query embedding and the memory embedding. Depending on the length limit, we select the top k memories with the highest rank scores as the activated memories. Here, the value of k can range from 3 to 10.

Use Summary To address the second question, we have designed a prompt to evaluate whether the user's question can be answered using the turn summary. We perform this evaluation for each activated memory that exceeds 800 tokens. It is important to note that the summary assessment occurs only when the total number of activation memory tokens exceeds 2000. If the assessment yields a positive result, indicating that the summary can answer the user's question, we use the memory summary to represent that specific memory.

3 MemoEval: The Dataset

To evaluate the effectiveness of the SCM framework in leveraging memories and processing lengthy inputs, we gather real-world data from real-world scenarios and subsequently train annotators to perform corresponding task labeling for long-term dialogues, book summarization, and meeting summarization. These tasks effectively illustrate the model's capacity to process lengthy inputs.

Table 1. The MemoEval dataset statistics. 2M means 2 million.

	Dialogue	**Book**	**Meeting**
#Instances	100	50	100
Avg tokens	30k	1.5M	36k
Max tokens	50k	2M	60k
Total tokens	10M	20M	15M
Max turn	200	-	80
Language	En+Zh	En+Zh	Zh

3.1 Dataset Construction

To guarantee the authenticity and relevance of our constructed dataset, the MemoEval dataset is developed by gathering data from real user scenarios, including (1) real user dialogues based on ChatGPT interactions from ShareChat[3], (2) a vast collection of open-source book data from the Gutenberg website[4], and (3) authentic multi-party meeting records from VCMeeting [28].

Using the following criteria, we apply data filtering to the three corpora mentioned above. Firstly, for the dialogue data, we eliminate any content that was deemed harmful, such as pornography, violence, and religious content. Secondly, we select classic literary works for the book data to honor the classics and facilitate annotation. Third, for the conference data, we use the open source VCMeeting corpus, which is in Chinese and includes the speech content of all participants in the conference, as well as the final summary of the entire conference. To ensure data quality, we remove content that consists of less than 20 turns.

In our approach to data annotation, we enlist and train a team of five annotators with distinct responsibilities. Firstly, the annotators must formulate five questions, assuming the role of the user, based on a multi-turn dialogue between the user and ChatGPT. These questions are required to integrate previously discussed topics or details and must be annotated with the corresponding answers and the index of supporting evidence. Secondly, when presented with a book text, the annotators identify a reference summary that encapsulates the central themes and key events or plots. Lastly, for conference data, reference summaries are already provided, obviating the need for additional annotation.

3.2 Data Statistics and Analysis

We present comprehensive statistics and analysis results for the dataset in Table 1. The MemoEval dataset is divided into three categories: Dialogue, book and

[3] `https://paratranz.cn/projects/6725`
[4] `https://www.gutenberg.org/`

meeting. The dataset comprises 100 instances for dialogue, 50 instances for book, and 100 instances for meetings. In terms of token count, the average number of tokens for Dialogue is 30,000, while for Book and Meeting, the averages are 1.5 million and 36,000, respectively. The dataset includes content in both English and Chinese for dialogue and book, whereas the Meeting category contains only data in Chinese.

Table 2. Long-term dialogue evaluation results. The lower part of the table is the ablation experiment of our framework.

Model	Answer Acc.	Memory Retrieval Recall	Single-Hop Acc.	Multi-Hop Acc.
ChatGPT	30.6	–	31.8	22.5
SCM $_{\text{turbo}}$ (ours)	68.3	93.5	73.5	64.3
SCM $_{\text{davinci003}}$ (ours)	**77.1**	**94.0**	**79.6**	**75.0**
w/o memory controller	59.3 (-17.8)	93.8 (-0.2)	71.7 (-7.9)	49.4 (-25.6)
w/o flash memory	72.9 (-4.2)	93.9 (-0.1)	74.6 (-5.0)	74.8 (-0.2)
w/o activation memory	50.4 (-26.7)	0.0 (-94.0)	30.2 (-49.4)	10.7 (-64.3)

4 Experiments

To evaluate the effectiveness and robustness of the SCM framework, we perform extensive experiments on three tasks: long-term dialogues, book summarization, and meeting summarization. Then, we investigate whether memory-enhanced LLMs can offer more comprehensive coverage and create coherent contextual logic summaries compared to traditional LLMs when tackling long-text summarization scenarios.

4.1 Baselines

To ensure a fair comparison, we have selected specific model variants for experimental analysis:

- ChatGPT: Vanilla ChatGPT api *gpt-3.5-turbo-0301*, only keeping the content of the context window length.
- SCM $_{\text{turbo}}$: Using *gpt-3.5-turbo-0301* as the backbone of our SCM framework.
- SCM $_{\text{davinci003}}$: Using *text-davinci-003* as the backbone for the SCM framework.
- SCM $_{\text{davinci003}}$ *w/o* memory controller: Remove the memory controller and concatenate the full retrieved content. If the token length of the concatenated history exceeds the context window, then truncate it.
- SCM $_{\text{davinci003}}$ *w/o* flash memory: Remove flash memory (short-term memory) which contains the latest information.

– SCM $_{\text{davinci003}}$ *w/o* activation memory: Remove activation memory (long-term memory), which is essential for answering questions involving long-distance dependencies.

4.2 Evaluation Metrics

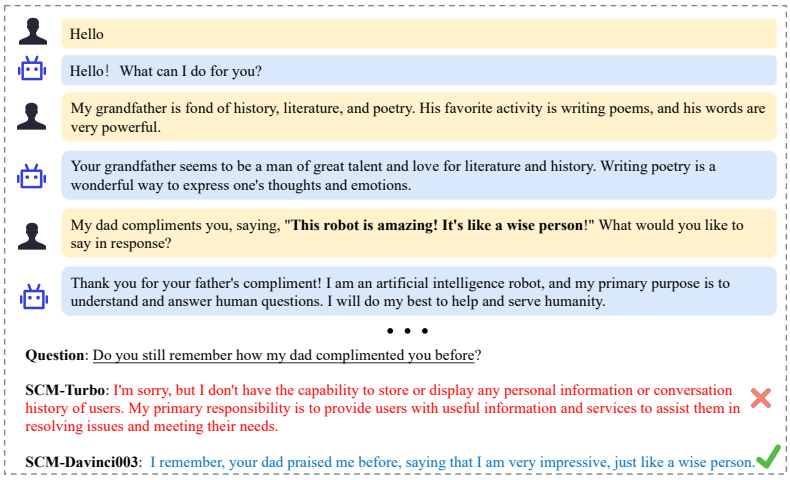

Fig. 4. The result comparison between SCM $_{\text{davinci003}}$ and SCM $_{\text{turbo}}$ in the long-term dialogue task.

Distinct evaluation metrics are utilized for long-term dialogue tasks and summarization tasks. For long-term dialogue tasks, the performance of our framework is assessed based on the following metrics. (1) Answer Accuracy, testing the accuracy of all probing questions. (2) Memory Retrieval Recall, which evaluates the performance of a memory controller. (3) Single-Hop Accuracy, evaluating the accuracy of single-hop-related probing questions. (4) Multi-Hop Accuracy, examining the accuracy of multi-hop-related probing questions. For the summarization task, two metrics, coverage and coherence, are used to evaluate content coverage and plot coherence as below:

$$coverage = f(\text{Summary}, \text{Source}) \tag{1}$$

$$coherence = g(\text{Summary}) \tag{2}$$

where f represents a function that calculates coverage by comparing the summary with the source material, and g represents a function that calculates the coherence of the summary. To facilitate a comprehensive comparison, we assess the effectiveness of the model by comparing its win rate to that of the baseline model, namely RecursiveSum [29] by OpenAI, which first summarizes small sections of the book and then recursively summarizes these summaries to produce a summary of the entire book.

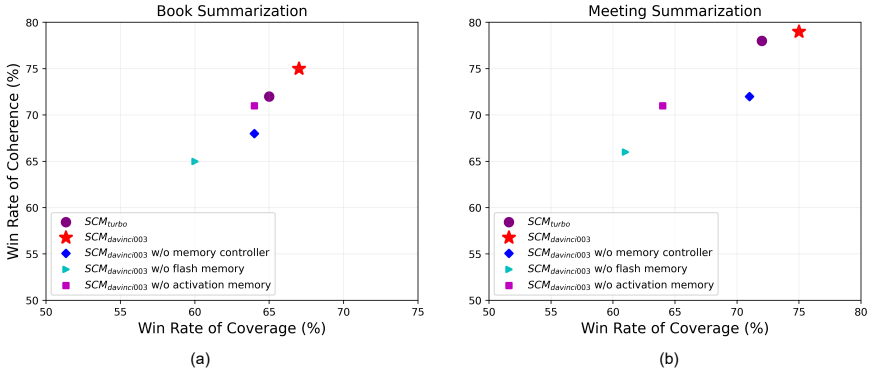

Fig. 5. The win rate of SCM variants against baseline model, RecursiveSum [29] by OpenAI, in both book and meeting summarization tasks. The figure also shows a comparison of the results of the SCM framework and its various component ablations.

4.3 Main Results

To quantitatively compare the models' performance, 100 test questions are annotated based on the dialogue data and categorized into two groups: single-hop-related questions and multi-hop-related questions. In addition, we compare the performance of SCM variants with the baseline model to evaluate the two summarization tasks.

Dialogue Results Table 2 displays the results of the long-term chatGPT dialogue and the variant method of the SCM framework. Our SCM methods have achieved a 40% improvement in answer accuracy compared to the vanilla ChatGPT. This improvement is attributed to the limitations of the vanilla ChatGPT, which processes data only within a fixed context window length. This results in the loss of information beyond this limit and, consequently, incorrect responses. In contrast, SCM-based methods preserve extensive dialogue histories in a memory stream, enabling the retrieval of the most relevant memories when necessary. These memories are then used to generate the final response. Furthermore, SCM $_{\text{davinci003}}$ exceeds SCM $_{\text{turbo}}$ by 9% in terms of the accuracy of the response. This may be attributed to the conservative nature of SCM $_{\text{turbo}}$, which can lead to hesitation in answering privacy-related probing questions. In contrast, SCM $_{\text{davinci003}}$ is capable of providing quicker and more precise responses.

Summarization Results A side-by-side comparison is performed by human annotators to check the summarization ability of our framework. Annotators must choose which one is better based on the answers. They are blind to models and other information. Figure 5 illustrates the results of the book and the summary of the meeting. Based on the experimental results, we have obtained the following conclusions: (1) SCM $_{\text{davinci003}}$ provides better coverage than SCM $_{\text{turbo}}$. (2)

SCM $_{\text{davinci003}}$ and SCM $_{\text{turbo}}$ demonstrate comparable coherence performance due to their memory-enhanced mechanism. (3) The SCM framework without memory loses contextual dependency and consequently produces unsatisfactory summarization outcomes. It is evident from the model comparison results that SCM $_{\text{davinci003}}$ consistently outperforms SCM $_{\text{turbo}}$ summarizing both books and meetings. This can be attributed to the fact that the summarization of SCM $_{\text{turbo}}$ focuses primarily on general principles, whereas it overlooks detailed core plots. In terms of human evaluation, the results of the SCM $_{\text{davinci003}}$ model are favored more because of their conciseness, clarity, and richer plot content.

Ablation Results We conduct an ablation study to investigate the independent effect of each module in the SCM framework, the results are illustrated in the lower part of Table 2. When activation memory is removed, the accuracy of the framework's responses significantly decreases, resulting in an approximate 60% decrease in performance. This is because most probing questions are derived from long-distance dialogue records, which rely on activation memory to retrieve them. The removal of activation memory from the model has a dramatic effect, entirely eliminating its ability to perform memory retrieval recall and to answer multi-hop questions correctly. This highlights the crucial role of activation memory in model functionality. Moreover, discarding the memory controller has a disproportionately adverse effect on multi-hop question accuracy when compared to single-hop questions. The reason is that the memory controller is responsible for dynamically filtering memory and utilizing summaries to effectively manage input tokens.

4.4 Further Analysis

The purpose of this qualitative study is to answer three research questions (RQs). The following experiment evaluates the performance of the SCM $_{\text{davinci003}}$ model without dialogue optimization compared to the vanilla ChatGPT model.

> **RQ1**. Can the SCM framework compete with or even outperform ChatGPT within its context window?

The example in Figure 1 includes 16k tokens, in which the user asked about his hobbies and discussed 100+ turns ago with the agent. ChatGPT fails to answer this question primarily because it is distracted by a significant amount of irrelevant historical noise. In contrast, our SCM framework can accurately respond to the query, demonstrating its superiority over vanilla ChatGPT.

> **RQ2**. Can the SCM framework scale to provide accurate responses to user questions, which are related to historical contexts that date back hundreds or even thousands of turns?

The example presented in Figure 6 illustrates a long-term dialogue comprising over 500 turns. At the outset, the user states that his goal is to reduce weight and intends to initiate a running regime. Subsequently, the user and the model chat daily about progress toward achieving their weight loss goals, among other

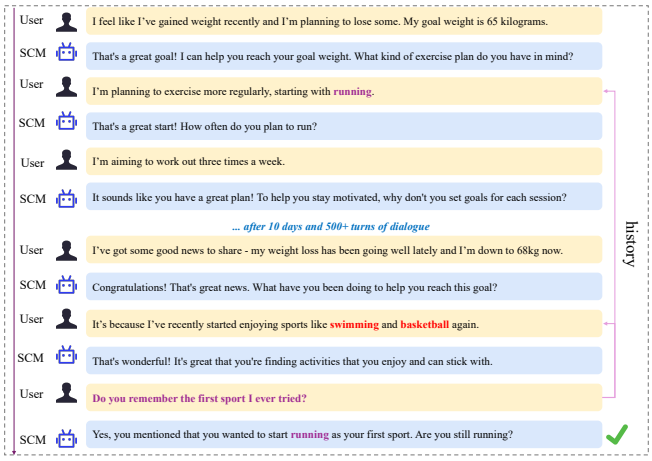

Fig. 6. Long-term dialogue example. To answer users' questions, our model can accurately retrieve relevant memories from massive memories and generate accurate responses based on these memories.

discussion topics. After more than 500 rounds of dialogue, the token length of the conversation has already exceeded 16k tokens. The user then asks the model "Do you remember the first sport I tried?". Our SCM framework recalls sports-related information from memory and combines it with the user's current question and generates an accurate response. These results suggest that the SCM framework can effectively scale with increasing numbers in memory.

RQ3. Can SCM demonstrate effective generalization to other lengthy input scenarios?

Figure 7 illustrates an example of summarizing lengthy books and meetings with our SCM framework iteratively and hierarchically. This lengthy document has been divided into several parts, and it has been gradually summarized to obtain the first-level local summary, and then hierarchically summarized to obtain the final summary. To maintain context coherence, relevant memories from previous sections will be added to the input text. The conventional method involves dividing lengthy texts into separate smaller text blocks that can be processed by the model. and summarizing each text block independently. However, this method can lose the dependency relationship between paragraphs. Our SCM framework facilitates the summarization process by utilizing the related memories, thus establishing substantial coherence between the two summaries. Ultimately, the framework incorporates a divide-and-conquer strategy to generate the final summary of the document. These results demonstrate the generalizability of the SCM framework in long input scenarios.

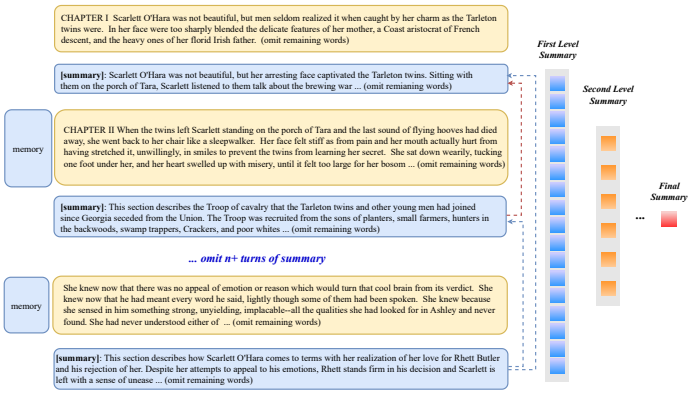

Fig. 7. An book summarization example from *Gone With The Wind* with SCM framework. Our framework divides the text into small blocks and sequentially summarizes each block. We then hierarchically summarize the first-level summary until reaching the final summary.

5 Related Work

5.1 Large Language Models

Large Language Models (LLMs) are language models trained on massive amounts of text data [23, 10, 30] based on the Transformer architecture. The pre-training and fine-tuning paradigm has contributed to several downstream language understanding and generation tasks. Subsequently, GPT-1 [20], GPT-2 [21], and GPT-3 [7] are developed with gradually increasing parameter sizes (GPT-3 has 175B parameters). LLMs enhanced by instruction tuning have shown emergent abilities in complex reasoning [26, 27, 8], knocking both academia and industry. Despite their advancements, these models exhibit significant limitations. A prominent deficiency is their lack of a long-term memory mechanism, which hinders their ability to maintain context over extended periods and retrieve relevant information from previous interactions. This paper addresses this issue by aiming to develop a self-controlled memory mechanism for LLMs.

5.2 Long Text Sequence Processing

Handling long text sequences has been a persistent challenge in natural language processing tasks [3, 24, 18, 2]. Existing solutions primarily involve replacing the Attention structure during pre-training to reduce computational costs and expanding the pre-training sequence length [5, 31, 12]. Another alternative approach [19] uses special positional encoding during pre-training to enable the model to learn relative positions and handle longer input texts during inference. These methods includes Position Interpolation [9, 19], NTK-aware [1] and Yarn [17]. However, the generalizability of these methods and their impact on downstream tasks remain uncertain. In the field of long-text summarization,

there are many effective methods. Hierarchical or iterative methods have been used by [29, 33, 34] to handle long texts by decomposing a complex problem into multiple sub-problems. However, these methods fail to capture the relationships among sub-problems.

5.3 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) enhances LLMs by retrieving relevant document chunks from the external knowledge base through semantic similarity calculation [11]. The primary purpose of this technology is to address the hallucination and outdated information issues of LLMs, particularly in domain-specific or knowledge-intensive tasks. Naive research on RAG follows a traditional process that includes indexing, retrieval, and generation [14]. Indexing starts with cleaning and extracting raw data in various formats. This step is crucial to enable efficient similarity searches in the subsequent retrieval phase. Advanced RAG introduces specific improvements to overcome the limitations of naive RAG, including pre-retrieval and post-retrieval strategies. The primary difference between our method and the conventional RAG approach is its emphasis on automating memory management control. LLM agents fully drive the decision-making process of SCM.

6 Conclusion and Future Work

In this paper, we propose the self-controlled memory (SCM) framework, designed to enhance the capacity of LLMs to maintain long-term memory and retrieve relevant information. Furthermore, we release the MemoEval evaluation dataset to evaluate the efficacy of SCM in memory utilization and processing of long inputs. We contribute the MemoEval dataset to evaluate the effectiveness of SCM in three tasks: long-term dialogues, book summarization, and meeting summarization. Experimental results indicate that the SCM framework empowers LLMs with infinite memory, thereby efficiently addressing the problem of forgetting historical information. Moreover, when equipped with the SCM framework, LLMs show the ability to process data far beyond the length of their context window while maintaining considerable performance. Experimental results demonstrate that our SCM framework significantly enhances overall accuracy by nearly 40% compared to vanilla ChatGPT in the long-term dialogue task. We investigate the effectiveness of the SCM utilizing two close-sourced models from OpenAI. Future research will involve conducting extensive evaluations of SCM performance using open-source, long-context window models.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 62276017, 62406033, U1636211, 61672081), and the State Key Laboratory of Complex& Critical Software Environment (Grant No. SKLCCSE-2024ZX-18).

References

1. Ntk-aware scaled rope (2023), https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/
2. Bai, J., Guo, H., Liu, J., Yang, J., Liang, X., Yan, Z., Li, Z.: Griprank: Bridging the gap between retrieval and generation via the generative knowledge improved passage ranking (2023)
3. Bai, J., Yang, Z., Yang, J., Guo, H., Li, Z.: Kinet: Incorporating relevant facts into knowledge-grounded dialog generation. IEEE/ACM Transactions on Audio, Speech, and Language Processing (2023)
4. Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Yu Han, e.a.: Qwen technical report (2023)
5. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer (2020)
6. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Pranav Shyam, e.a.: Language models are few-shot learners. In: Proc. of NeurIPS (2020)
7. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners (2020)
8. Chai, L., Yang, J., Sun, T., Guo, H., Liu, J., Wang, B., Liang, X., Bai, J., Li, T., Peng, Q., et al.: xcot: Cross-lingual instruction tuning for cross-lingual chain-of-thought reasoning. arXiv preprint arXiv:2401.07037 (2024)
9. Chen, S., Wong, S., Chen, L., Tian, Y.: Extending context window of large language models via positional interpolation. CoRR **abs/2306.15595** (2023)
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. of NAACL (2019)
11. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey (2024)
12. Guo, M., Ainslie, J., Uthus, D., Ontanon, S., Ni, J., Sung, Y.H., Yang, Y.: LongT5: Efficient text-to-text transformer for long sequences. In: Proc. of ACL Findings (2022)
13. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts (2023)
14. Ma, X., Gong, Y., He, P., Zhao, H., Duan, N.: Query rewriting for retrieval-augmented large language models. arXiv preprint arXiv:2305.14283 (2023)
15. Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., Gao, J.: Large language models: A survey (2024)
16. OpenAI: Gpt-4 technical report (2024)
17. Peng, B., Quesnelle, J., Fan, H., Shippole, E.: Yarn: Efficient context window extension of large language models. CoRR **abs/2309.00071** (2023)
18. Pi, X., Wang, B., Gao, Y., Guo, J., Li, Z., Lou, J.G.: Towards robustness of text-to-sql models against natural and realistic adversarial table perturbation (2022)
19. Press, O., Smith, N., Lewis, M.: Train short, test long: Attention with linear biases enables input length extrapolation. In: Proc. of ICLR (2022)

20. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Improving language understanding with unsupervised learning (2018)
21. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
22. Stiennon, N., Ouyang, L., Wu, J., Ziegler, D.M., Lowe, R., Voss, C., Radford, A., Amodei, D., Christiano, P.F.: Learning to summarize from human feedback. CoRR (2020)
23. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Proc. of NeurIPS (2017)
24. Wang, B., Gao, Y., Li, Z., Lou, J.G.: Know what i don't know: Handling ambiguous and unanswerable questions for text-to-sql (2023)
25. Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M., Le, Q.V.: Finetuned language models are zero-shot learners. In: Proc. of ICLR (2022)
26. Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E.H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., Fedus, W.: Emergent abilities of large language models. TMLR (2022)
27. Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E.H., Le, Q.V., Zhou, D.: Chain of thought prompting elicits reasoning in large language models. In: Proc. of NeurIPS (2022)
28. Wu, H., Zhan, M., Tan, H., Hou, Z., Liang, D., Song, L.: Vcsum: A versatile chinese meeting summarization dataset (2023)
29. Wu, J., Ouyang, L., Ziegler, D.M., Stiennon, N., Lowe, R., Leike, J., Christiano, P.: Recursively summarizing books with human feedback (2021)
30. Yang, J., Ma, S., Huang, H., Zhang, D., Dong, L., Huang, S., Muzio, A., Singhal, S., Hassan, H., Song, X., Wei, F.: Multilingual machine translation systems from microsoft for WMT21 shared task. In: WMT@EMNLP 2021, Online Event, November 10-11, 2021 (2021)
31. Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., Ahmed, A.: Big bird: Transformers for longer sequences (2021)
32. Zeng, A., Liu, X., Du, Z., Wang, Z., Lai, H., Ding, M., Yang, Z., Xu, Y., Zheng, W., Xia, X., Tam, W.L., Ma, Z., Xue, Y., Zhai, J., Chen, W., Liu, Z., Zhang, P., Dong, Y., Tang, J.: GLM-130b: An open bilingual pre-trained model. In: Proc. of ICLR (2023)
33. Zhang, Y., Ni, A., Mao, Z., Wu, C.H., Zhu, C., Deb, B., Awadallah, A., Radev, D., Zhang, R.: Summ^n: A multi-stage summarization framework for long input dialogues and documents. In: Proc. of ACL (2022)
34. Zhong, W., Guo, L., Gao, Q., Ye, H., Wang, Y.: Memorybank: Enhancing large language models with long-term memory (2023)