Enhancing Long Context Performance in LLMs Through Inner Loop Query Mechanism

Yimin Tang[1,2], **Yurong Xu**[1], **Ning Yan**[1], **Masood Mortazavi**[1]
{ytang4,yurong.xu,yan.ningyan,masood.mortazavi}@futurewei.com
[1]Futurewei Technologies Inc, Santa Clara, 95050
[2]University of Southern California, Los Angeles, 90089

Abstract

Transformers have a quadratic scaling of computational complexity with input size, which limits the input context window size of large language models (LLMs) in both training and inference. Meanwhile, retrieval-augmented generation (RAG) besed models can better handle longer contexts by using a retrieval system to filter out unnecessary information. However, most RAG methods only perform retrieval based on the initial query, which may not work well with complex questions that require deeper reasoning. We introduce a novel approach, Inner Loop Memory Augmented Tree Retrieval (ILM-TR), involving inner-loop queries, based not only on the query question itself but also on intermediate findings. At inference time, our model retrieves information from the RAG system, integrating data from lengthy documents at various levels of abstraction. Based on the information retrieved, the LLM generates texts stored in an area named Short-Term Memory (STM) which is then used to formulate the next query. This retrieval process is repeated until the text in STM converged. Our experiments demonstrate that retrieval with STM offers improvements over traditional retrieval-augmented LLMs, particularly in long context tests such as Multi-Needle In A Haystack (M-NIAH) and BABILong.

1 Introduction

Large Language Models (LLMs) have demonstrated a powerful ability to handle almost all kinds of NLP tasks with impressive performance [7, 1, 29]. As the size of LLMs increases, they tend to perform better and store more informative knowledge within their parameters [17, 25]. LLMs can also be further fine-tuned on downstream tasks [34]. However, the length of the input window in LLMs is constrained by the quadratic computational complexity of the self-attention mechanism, which is a fundamental structure in these models [31]. An alternative approach to processing longer contexts is to split large quantities of text into chunks and index these chunks as vectors in a separate information retrieval system [20, 5, 14]. Since the retrieval system can filter out unnecessary information, the LLM can handle user questions with long raw data within a limited context window. Additionally, this approach provides easier interpretability and provenance tracking compared to the opaque and unexplainable parameters within LLMs [2].

However, existing retrieval-augmented approaches also have shortcomings. The issue we aim to address is that most existing methods retrieve only a few text chunks that are directly related to user queries, which limits the ability of LLMs to produce deeper answers to questions. This is particularly relevant when it comes to fully understanding or integrating knowledge from multiple parts that may not be directly related to the user's questions, such as understanding foreshadowing in novels or inferring the identity of the killer in detective fiction. When humans perform such tasks, we often have impressions in our minds related to unusual facts that may connect to our questions, indicating that memory plays a crucial role in advanced comprehension skills [6, 19]. Long-term and short-term

Accepted as a Workshop Paper at NeurIPS 2024 Workshop on Adaptive Foundation Models

memory are essential aspects of human-like intelligence, particularly in maintaining an understanding of long contexts, such as lifelong conversations where recalling past interactions is crucial for rapport building and long context comprehension. Some interesting works have been explored [32, 37] and implemented in real products, such as memory in medical applications [36] and ChatGPT's memory capabilities [24]. However, these works still rely on a single retrieval query in the memory system and lack a mechanism that can automatically gather more information based on the current context, making it difficult to accurately answer user questions.

To address this, we designed an inner-loop mechanism that allows retrieval to be conditioned not only on the initial query but also on the current information obtained. Our system, Inner Loop Memory Augmented Tree Retrieval (ILM-TR), leverages the retrieved information to generate intermediate answers, which are then used to formulate subsequent queries. This inner-loop query continues until the answer converges or the query limit is reached. This mechanism can effectively answer complex questions in long context scenarios.

Our main contributions are as follows: 1) We propose a novel summarization method which not only summarizes the chunked texts but also outputs all surprising facts within the content. 2) We propose an inner-loop mechanism which refines the retrieval process by conditioning it not only on the user's initial query but also on the evolving information gathered during the retrieval process. Our approach improves comprehension in long-context scenarios by enabling more effective information retrieval and interpretation. 3) we test the ILM-TR system on standard long-context benchmarks such as Multi-Needle In A Haystack (M-NIAH) [16], and BABILong [18] with Llama3 as our summary model and answer model. Experimental results demonstrate that our method outperforms baseline RAG method and maintains robust performance, with no significant degradation even as the context length scales up to 500k tokens.

2 Related Work

2.1 Large Language Models (LLMs)

LLMs such as the GPT [7, 1], Gemini [29, 10], and Claude [3] series have made remarkable strides across a broad spectrum of tasks and have increasingly become daily assistants for many people. However, the closed-source nature of these models prohibits researchers and companies from studying the inner mechanisms of LLMs and building domain-adapted applications. Consequently, many open-source LLMs have emerged in the community, such as Llama [30, 9], ChatGLM [12], and Mistral [15]. However, these models still have limited context windows, usually capped at 8k tokens, due to the complexity of self-attention and position encoding. The limited context windows imply that they lack long-term memory capabilities, which could be enhanced by integrating RAG systems with memory structures.

2.2 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) uses retrieved tokens from long context raw data to extend the input window size of LLMs. The original RAG [20] integrates pre-trained sequence-to-sequence models with a neural retriever. [23] introduced the Joint Passage Retrieval (JPR) model, which employs a tree-decoding algorithm to handle passage diversity and relevance in multi-answer retrieval. Dense Hierarchical Retrieval (DHR) and Hybrid Hierarchical Retrieval (HHR) represent advancements in retrieval accuracy by combining document-level and passage-level retrievals and integrating sparse and dense retrieval methods, respectively [21, 4]. RAPTOR [27] utilizes clustering and summarizing of text chunks, constructing a tree with varying levels of summarization from the bottom up, enabling multiple levels of understanding of long contexts. Nevertheless, these approaches still depend on a single retrieval query and do not include a mechanism for automatically acquiring additional information based on the evolving context, which makes it challenging to provide precise answers to user questions.

2.3 Memory Mechanisms

Efforts have been made to improve the memory capabilities of neural models. Memory-augmented networks (MANNs) [22], such as Neural Turing Machines (NTMs) [13], enhance the memory capacity of neural networks by incorporating an external memory matrix, allowing them to manage tasks that

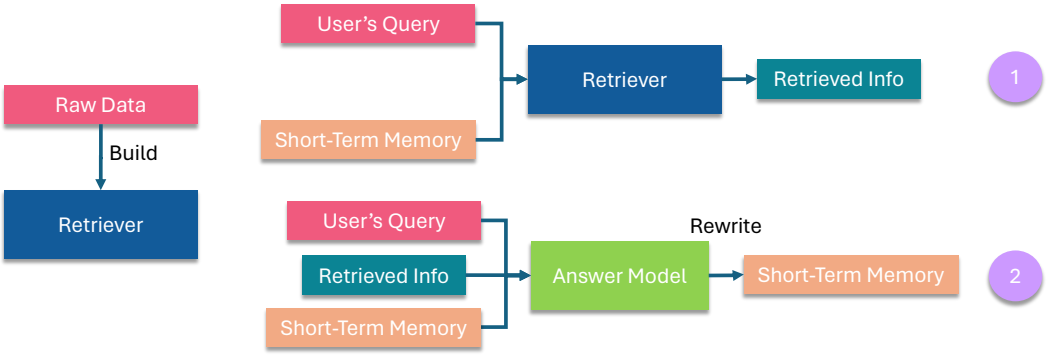


Figure 1: An Overview of **ILM-TR** Method. **Raw Data** consists of tokens from the user, which could include conversation history, novels, or any other content the user wants the LLMs to process. **User's Query** refers to the tokens provided by the user, such as questions or task descriptions. **Retriever** can be any retrieval method, such as sentence-based RAG or tree-structured RAG. **Retrieved Info** is the result produced by the retriever. **Short-Term Memory** is a storage area for a limited number of tokens, which is overwritten at each iteration of the inner-loop query. **Answer Model**(LLMs) will processe information from the retriever, the previous short-term memory, and the user's query. The purple circles represents the order of steps in the inner-loop query.

require the storage and manipulation of information over extended periods. [28] demonstrates that LLMs with external memory can simulate Turing Machines. Although promising, these approaches have yet to fully address the need for a dependable and adaptable memory function in LLMs. Some studies have focused on long-range conversations [33, 35], but these are typically limited to a few conversational rounds, falling short of supporting long-term AI companions. Additionally, these models often struggle to create detailed user profiles and lack a human-like memory updating mechanism, both of which are essential for enabling more natural interactions.

3 Method

Our Inner Loop Memory-Augmented Tree Retrieval (ILM-TR) method, as shown in Fig. 1, contains two parts: retriever and inner-loop query. For the retriever part, we primarily use the RAPTOR's tree-build method [27]. The retriever first segments the raw data into short, contiguous text chunks of a certain length. If a sentence exceeds the length limit, it will be moved to the next chunk. After splitting, an summary model is used to summarize each chunk. However, unlike typical summarization methods such as RAPTOR, our model produces two kinds of summaries: one is the regular summary of the main text in the chunk, while the other includes all surprising facts that differ from the main text. The retriever architecture is illustrated in Figure 2.

Building upon the idea that the informational value of a communicated message depends on the degree of surprise in its contents [8], the inclusion of surprising information, distinct from the main text, will also provide valuable insights to LLMs when handling long contexts. After generating summary texts and surprising information from each chunk, we group similar texts using Gaussian Mixture Models, as employed in RAPTOR (refer to [27] for more details). However, we only group the summaries without the surprising information.

All texts are embedded for searching and clustering using SBERT, a BERT-based encoder (multi-qa-mpnet-base-cos-v1) [26]. These summarized texts are then re-embedded, and the cycle of embedding, clustering, and summarization continues until further clustering becomes impractical, resulting in a structured, multi-layered tree representation of the raw data. For querying within this tree, similar to RAPTOR, we use a collapsed tree strategy that disregards the tree structure and directly traverses all the nodes as shown in Figure 2.

For inner-loop query part, as shown in Fig. 1, we use an LLM as the answer model to generate the final answer. This model can be the same as the summary model or a separate one. We create an additional area called Short-Term Memory (STM), which stores texts up to the answer model maximum output length. The STM is initially empty. Each time, the answer model generates an

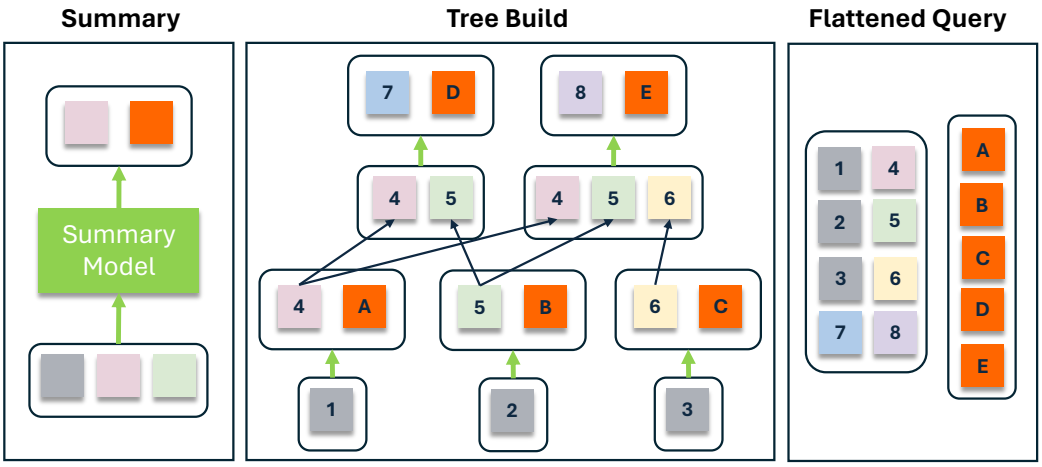


Figure 2: The **ILM-TR** Retriever: The orange square represents the surprising information, the grey color represents the original text, and other colors represent the summary information. The summary model will extract information from the provided tokens. During the tree-building process, the summary information will be grouped using a clustering algorithm, and then each group will be summarized together to generate a higher-level summary and surprising information. In the query process, all squares in the tree will be stored in a table, and the best fit will be returned based on vector distance from the query text.

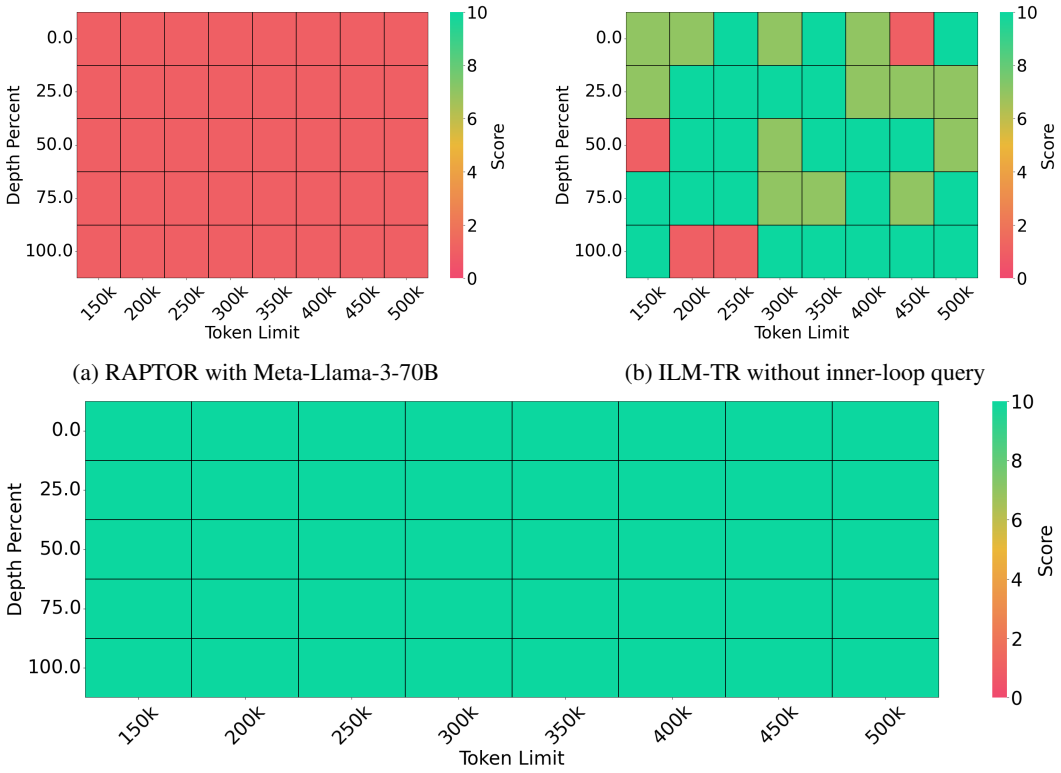
answer based on the user's query, information retrieved by the retriever, and the previous texts in the STM. Then the answer will be stored in the STM. The retriever then uses the contents of the STM along with the user's query to retrieve new information from the raw data. Once new information is retrieved, the answer model generates new texts and stores them in the STM. This process is repeated in the inner-loop query until the texts in the STM converge and stop changing, or until the query limit is reached. Finally, the STM texts are returned to the user as the final answer.

4 Experiments

We evaluate ILM-TR's long-context performance using two benchmarks: M-NIAH [16] and BABILong [18]. For both the summary and answer model inference, we utilize Meta-Llama-3-70B with llama.cpp [11], quantized using Q4_K_M due to hardware limitations. We do not employ smaller LLMs, such as the 8B or 7B models, as their instruction-following capabilities were found to be inadequate in our tests: they consistently failed to follow the summarization prompts correctly. We also set the maximum number of inner-loop queries to 5. All tests were conducted on a machine running Ubuntu 22.04, equipped with an Intel Xeon Gold 6242 processor and four NVIDIA Tesla V100 32GB GPUs. Details of all prompts and parameters are provided in Appendices A.1 and A.3.

In the M-NIAH test, several sentences are inserted into a specific area of a given long context. The question is related to all the inserted sentences, and the model is expected to retrieve all necessary information across these sentences. For example, we use three sentences: 'Figs are one of the secret ingredients needed to build the perfect pizza', 'Prosciutto is one of the secret ingredients needed to build the perfect pizza' and 'Goat cheese is one of the secret ingredients needed to build the perfect pizza'. The question would then be 'What is the first letter of each secret ingredient needed to build the perfect pizza?'. The BABILong test is similar to the M-NIAH test but involves sentences with more complex logical relationships. For instance, it may include sentences like 'The apple is in the bathroom' and 'Jack takes the apple to the kitchen' The question in this case would be: 'Where is the apple before kitchen?'.

We present the M-NIAH and BABILong test results in Figure 3 and Figure 4(Appendix A). We tested RAPTOR as the baseline method, and our ILM-TR method with two settings, with token lengths ranging from 150k to 500k. There are three inserted sentences for M-NIAH test. Each testcase has four possible score levels: no keywords found (score 1, red), one keyword found (score 3, orange), two keywords found (score 7, yellow), and all keywords found (score 10, green). Figure 3 demonstrates



(a) RAPTOR with Meta-Llama-3-70B

(b) ILM-TR without inner-loop query

(c) ILM-TR

Figure 3: M-NIAH test: no keywords found (score 1, red), one keyword found (score 3, orange), two keywords found (score 7, yellow), and all three keywords found (score 10, green). Token lengths range from 150k to 500k. Depth percent represents the average positions of the inserted sentences within the long text, where 0% indicates the beginning of the text and 100% indicates the end.

that incorporating surprising information can significantly improve the model's performance in the M-NIAH test. However, there are still some cases where ILM-TR without the inner-loop query cannot retrieve all the keywords in a single query. The ILM-TR method, with inner-loop query capabilities, shows its ability to locate all the keywords in the M-NIAH test. In the BABILong test, our ILM-TR method also shows significant improvements compared to the baseline method. We show an simple example for BABILong with our ILM-TR model in Appendix A.2.

However, there are some shortcomings with ILM-TR. First, the inner-loop process requires several iterations, which increases the overall time consumption of query processing. Second, since we incorporate surprising information during summarization, the summary model must be capable of following complex instructions, which led us to choose a larger model with slower inference performance.

5 Conclusion and Future Work

We introduce a novel approach, Inner Loop Memory Augmented Tree Retrieval (ILM-TR), which incorporates inner-loop queries based not only on the initial query but also on intermediate findings. During inference, ILM-TR retrieves information from the RAG system. Based on the retrieved information, ILM-TR generates text that is stored in Short-Term Memory (STM), which is then used to formulate subsequent queries. This retrieval process is repeated until the text in STM converges. Our experiments demonstrate that retrieval with STM offers improvements over traditional retrieval-augmented LLMs in the M-NIAH and BABILong test. And since the answers for M-NIAH tests are known, in future work, we can explore fine-tuning the answer model output based on the intermediate results from STM and the reference answer. This could potentially improve model's active search capabilities with RAG system.

References

[1] Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

[2] Akyürek, E.; Bolukbasi, T.; Liu, F.; Xiong, B.; Tenney, I.; Andreas, J.; and Guu, K. 2022. Towards tracing factual knowledge in language models back to the training data. *arXiv preprint arXiv:2205.11482*.

[3] Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. Accessed: 2024-08-19.

[4] Arivazhagan, M. G.; Liu, L.; Qi, P.; Chen, X.; Wang, W. Y.; and Huang, Z. 2023. Hybrid hierarchical retrieval for open-domain question answering. In *Findings of the Association for Computational Linguistics: ACL 2023*, 10680–10689.

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

[6] Blum, L.; and Blum, M. 2022. A theory of consciousness from a theoretical computer science perspective: Insights from the Conscious Turing Machine. *Proceedings of the National Academy of Sciences*, 119(21): e2115934119.

[7] Brown, T. B. 2020. Language models are few-shot learners. *arXiv preprint ArXiv:2005.14165*.

[8] El Gamal, A.; and Kim, Y.-H. 2011. *Network information theory*. Cambridge university press.

[9] et al., A. D. 2024. The Llama 3 Herd of Models. arXiv:2407.21783.

[10] et al., G. T. 2023. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv:2312.11805.

[11] Gerganov, G.; and other contributors. 2023. llama.cpp: Port of Facebook's LLaMA model in C/C++. `https://github.com/ggerganov/llama.cpp`. Accessed: 2024-10-03.

[12] GLM, T.; Zeng, A.; Xu, B.; Wang, B.; Zhang, C.; Yin, D.; Rojas, D.; Feng, G.; Zhao, H.; Lai, H.; et al. 2024. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. *arXiv preprint arXiv:2406.12793*.

[13] Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

[14] Izacard, G.; Lewis, P.; Lomeli, M.; Hosseini, L.; Petroni, F.; Schick, T.; Dwivedi-Yu, J.; Joulin, A.; Riedel, S.; and Grave, E. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 1(2): 4.

[15] Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

[16] Kamradt, G. 2024. LLMTest_NeedleInAHaystack. Accessed: 2024-08-19.

[17] Kaplan, J.; McCandlish, S.; Henighan, T.; Brown, T. B.; Chess, B.; Child, R.; Gray, S.; Radford, A.; Wu, J.; and Amodei, D. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

[18] Kuratov, Y.; Bulatov, A.; Anokhin, P.; Rodkin, I.; Sorokin, D.; Sorokin, A.; and Burtsev, M. 2024. BABILong: Testing the Limits of LLMs with Long Context Reasoning-in-a-Haystack. *arXiv preprint arXiv:2406.10149*.

[19] LeCun, Y. 2022. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1): 1–62.

[20] Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

[21] Liu, Y.; Hashimoto, K.; Zhou, Y.; Yavuz, S.; Xiong, C.; and Yu, P. 2021. Dense hierarchical retrieval for open-domain question answering. arXiv. *arXiv preprint arxiv:2110.15439*.

[22] Meng, L.; and Huang, M. 2018. Dialogue intent classification with long short-term memory networks. In *Natural Language Processing and Chinese Computing: 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8–12, 2017, Proceedings 6*, 42–50. Springer.

[23] Min, S.; Lee, K.; Chang, M.-W.; Toutanova, K.; and Hajishirzi, H. 2021. Joint passage ranking for diverse multi-answer retrieval. *arXiv preprint arXiv:2104.08445*.

[24] OpenAI. 2024. Memory and New Controls for ChatGPT.

[25] Petroni, F.; Rocktäschel, T.; Lewis, P.; Bakhtin, A.; Wu, Y.; Miller, A. H.; and Riedel, S. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

[26] Reimers, N. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084*.

[27] Sarthi, P.; Abdullah, S.; Tuli, A.; Khanna, S.; Goldie, A.; and Manning, C. D. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *arXiv preprint arXiv:2401.18059*.

[28] Schuurmans, D. 2023. Memory augmented large language models are computationally universal. *arXiv preprint arXiv:2301.04589*.

[29] Team, G.; Anil, R.; Borgeaud, S.; Wu, Y.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

[30] Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; Rodriguez, A.; Joulin, A.; Grave, E.; and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.

[31] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

[32] Wilmot, D.; and Keller, F. 2021. Memory and knowledge augmented language models for inferring salience in long-form stories. *arXiv preprint arXiv:2109.03754*.

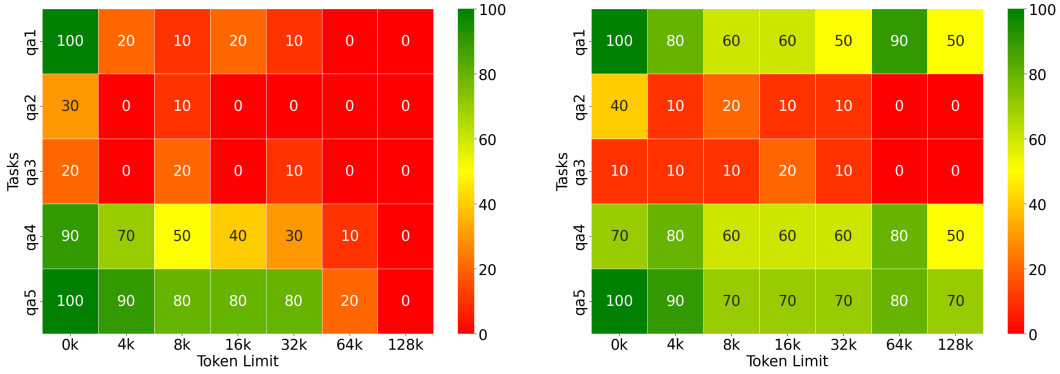
[33] Xu, J.; Szlam, A.; and Weston, J. 2021. Beyond goldfish memory: Long-term open-domain conversation. *arXiv preprint arXiv:2107.07567*.

[34] Xu, R.; Luo, F.; Zhang, Z.; Tan, C.; Chang, B.; Huang, S.; and Huang, F. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. *arXiv preprint arXiv:2109.05687*.

[35] Xu, X.; Gou, Z.; Wu, W.; Niu, Z.-Y.; Wu, H.; Wang, H.; and Wang, S. 2022. Long time no see! open-domain conversation with long-term persona memory. *arXiv preprint arXiv:2203.05797*.

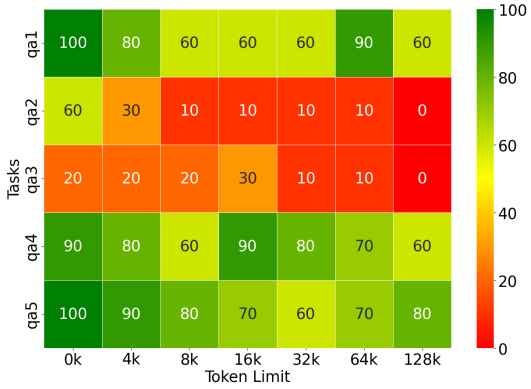
[36] Zhang, K.; Kang, Y.; Zhao, F.; and Liu, X. 2024. LLM-based Medical Assistant Personalization with Short-and Long-Term Memory Coordination. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2386–2398.

[37] Zhong, W.; Guo, L.; Gao, Q.; Ye, H.; and Wang, Y. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 19724–19731.



(a) RAPTOR with Meta-Llama-3-70B

(b) ILM-TR without inner-loop query



(c) ILM-TR

Figure 4: BABILong test: Due to hardware limitations and the size of the LLM model, we only tested 10 testcases for each test setting. In this study, we evaluated tasks qa1 to qa5 with token lengths ranging from 0k to 128k. We encourage readers to refer to [18] for further details.

A Appendix / supplemental material

A.1 Prompt

A.1.1 Baseline Prompt

Summary Model

[SYSTEM]: You are a reader who can summarize the given text while including important details. Do not provide any comments, just give the summary.

[USER]: Write a summary of the following context, just including the most important details: {context}

Answer Model

[SYSTEM]: You are Question Answering Portal.

[USER]: Given Context: {Retrieved Info} Give the best full answer to question {User's Query}

A.1.2 ILM-TR Prompt

Summary Model

[SYSTEM]: I will give you context. Most of it could be about the same things, but there may be some abnormal information. An surprising sentence is not related to most of the other content. You should summarize the context with the necessary information and also include any surprising information you think. Don't return any unrelated words.

Always and only return your answer with the following format, just list the fact based on given text, no comments, use different sentences to describe different facts:

(Summary): Your Summary

(Surprise): Surprising Information

[USER]: {context}

Answer Model without inner-loop

[SYSTEM]: You are Question Answering Portal.

[USER]: Given Context: {Retrieved Info} Give the best full answer to question {User's Query}

Answer Model with inner-loop

[SYSTEM]: You will be given some Retrieved Info and memory, and you will use this information to answer a question. If you can't answer the question, you can write something related to the question to help others answer it.
(Retrieved Info):
a2 = 1
(Memory):
a1 = a2+a3
(Question):
What is the value of a1
(Your Output):
a1 = a2 + a3. a2=1. We need to find the value of a3.

Keep your output short and don't return any unrelated words.

[USER]: {context} (Question): {User's Query}

A.2 BABILong Example with ILM-TR

A.2.1 Example (qa3, 128k)

[Needles]:
Daniel grabbed the milk.
Mary picked up the apple.
Sandra went back to the hallway.
Daniel journeyed to the hallway.
John moved to the bedroom.
John went to the bathroom.
Daniel discarded the milk there.
Mary moved to the kitchen.
Mary journeyed to the office.
Daniel got the milk.
John moved to the garden.
Sandra travelled to the kitchen.
Mary put down the apple.
John took the football.
[Question]: Where was the apple before the office?
[True Answer]: kitchen

A.2.2 ILM-TR Inner-loop Outputs

We calculate the Longest Common Subsequence (LCS) to determine whether the answer has converged.

[Round 1]:
The apple was at office. We need to find where the apple was before the office.
[Round 2]:
The apple was at office. Mary put down the apple at office. We need to determine

where Mary was before she placed the apple down.
[Round 3]:
The apple was at office. Mary put down the apple at office, but before that, she was in the kitchen.
[Round 4]:
Mary put down the apple at office, but before that, she was in the kitchen. The best answer to the question ""Where was the apple before the office?"" is:

The kitchen.
[Round 5]:
Based on the given context, the best answer to the question ""Where was the apple before the office?"" is:

The kitchen.

A.3 Model and Retriever Parameters

For parameters not mentioned here, the values are defualt values.

Parameter Category	Parameter Value
Answer Model	
temperature	0
frequency_penalty	1.2
max_tokens	200
Summary Model	
repeat_last_n	256
repeat_penalty	1.18
penalize_nl	False
presence_penalty	0
min_p	0.05
n_predict	1055
n_probs	0
mirostat	0
mirostat_eta	0.1
mirostat_tau	5
tfs_z	1
top_k	40
top_p	0.95
typical_p	1
frequency_penalty	0
temperature	0.2
Retriever	
tb_max_tokens (each chunk max token)	600
tb_summarization_length (output max token)	300