LONGAGENT: Achieving Question Answering for 128k-Token-Long Documents through Multi-Agent Collaboration

Jun Zhao[1*†], **Can Zu**[1*], **Hao Xu**[1], **Yi Lu**[1], **Wei He**[1], **Yiwen Ding**[1],
Tao Gui[3], **Qi Zhang**[1,2†], **Xuanjing Huang**[1,2]
[1]School of Computer Science, Fudan University
[2]Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
[3]Institute of Modern Languages and Linguistics, Fudan University
{zhaoj19,qz}@fudan.edu.cn,czu22@m.fudan.edu.cn

Abstract

Large language models (LLMs) have achieved tremendous success in understanding language and processing text. However, question-answering (QA) on lengthy documents faces challenges of resource constraints and a high propensity for errors, even for the most advanced models such as GPT-4 and Claude2. In this paper, we introduce LONGAGENT, a multi-agent collaboration method that enables efficient and effective QA over $128k$-token-long documents. LONGAGENT adopts a *divide-and-conquer* strategy, breaking down lengthy documents into shorter, more manageable text chunks. A leader agent comprehends the user's query and organizes the member agents to read their assigned chunks, reasoning a final answer through multiple rounds of discussion. Due to members' hallucinations, it's difficult to guarantee that every response provided by each member is accurate. To address this, we develop an *inter-member communication* mechanism that facilitates information sharing, allowing for the detection and mitigation of hallucinatory responses. Experimental results show that a LLaMA-2 7B driven by LONGAGENT can effectively support QA over $128k$-token documents, achieving 16.42% and 1.63% accuracy gains over GPT-4 on single-hop and multi-hop QA settings, respectively.

1 Introduction

Nowadays, the capabilities of large language models (LLMs) have been rapidly advancing, driven by ever-increasing model sizes and data volumes (OpenAI, 2023). However, the prohibitively high training costs preclude these capability gains from extending to LLMs' understanding of lengthy texts (Chen et al., 2023c). This poses a significant limitation on the practical applications of LLMs, such as querying information from books, analyzing legal documents or academic papers.

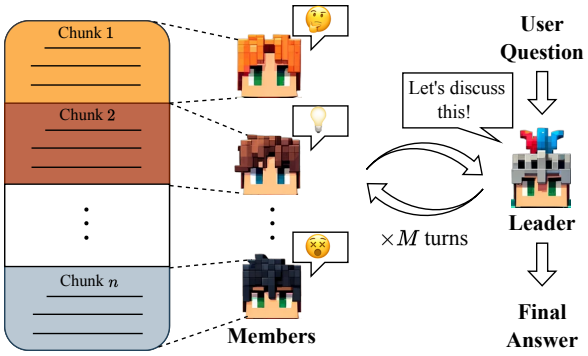

Figure 1: LONGAGENT collaboration scheme. The input long text (left) is segmented into chunks and assigned to corresponding members. The Leader receives user question (right), breaks them down into the simplest sub-question, convenes members for discussion, ultimately obtaining answers to all sub-question, and reasons to make the final response.

Researchers have primarily focused on two directions to address this issue. The first direction is to consider positional encoding as a crucial aspect (Press et al., 2022; Chen et al., 2023b; Peng et al., 2023; Chen et al., 2023a), taking techniques like extrapolation or interpolation to enable the positional encoding to handle *unseen* positions during pre-training. The second category employs more complex mechanisms like recurrent structures (Zhou et al., 2023; Zhang et al., 2024), token selection (Mohtashami and Jaggi, 2023; Tworkowski et al., 2023), or sliding windows (Xiao et al., 2023; Han et al., 2023) to enable the limited context window to process longer input texts. Despite these efforts, effectively understanding lengthy texts and accurately answering user queries remains a challenging problem (Hsieh et al., 2024). Models tailored for long text processing may compromise their innate capabilities for understanding shorter texts (Jin et al., 2024) and tend to overlook critical information situated in the middle of long documents, a phenomenon known

*Equal Contributions.
†Corresponding authors.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 16310–16324
November 12-16, 2024

as *lost in the middle* (Liu et al., 2023).

In this paper, we propose LONGAGENT, a novel approach for long-document QA. LONGAGENT employs a *divide-and-conquer* strategy, breaking down long documents into more manageable smaller text chunks. Leveraging the characteristic of aligned LLMs to follow instructions, LONGAGENT adopts a multi-agent collaboration approach to effectively support QA on ultra-long documents (over $100k$ tokens). *To alleviate conceptual confusion, we provide a detailed explanation in Appendix A that distinguishes* LONGAGENT *from long-context LLMs and multi-turn retrieval systems.* As shown in Figure 1, the agent team consists of a leader agent and multiple member agents. The leader is responsible for understanding the user's question and directing the members to gather clues from their assigned text chunks. Depending on the complexity of the question, this process may involve one or more rounds of interaction. The interaction ends when the leader deems the clues sufficient to reason the final answer. Managing a large number of members is non-trivial due to the model hallucination. We propose an *inter-member communication* mechanism to identify the members in a hallucinatory state and mitigate their influence to the leader's decision-making.

To comprehensively evaluate LONGAGENT, we propose *Needle-in-a-Haystack PLUS*.* Compared with original Needle-in-a-Haystack, it can test the model's capability of handling multi-hop QA tasks. We have also tested LONGAGENT with all long-document QA tasks of mainstream LongBench (Bai et al., 2023) and InfiniteBench (Zhang et al., 2023). The experimental results have demonstrated the effectiveness of LONGAGENT.

The contributions of this paper are as follows: 1) we propose LONGAGENT, which enables $4k$ context-driven LLMs to achieve QA on $128k$ long documents; 2) we develop the *Needle-in-a-Haystack PLUS*, which enables comprehensive evaluation of LLMs' capabilities in long-document QA. 3) our experimental results show that the LLaMA2-7B model driven by LONGAGENT exhibits superior long-text QA capabilities comparable to GPT-4.

*https://github.com/zuucan/NeedleInAHaystack-PLUS

2 LONGAGENT for Long-Document QA

2.1 Method Overview

We first formulate the problem that LONGAGENT aims to solve. Given a document $d = \{w_i\}_{i=1}^{n}$ and a user's question q, our goal is to build a QA model that can respond to the question according to the content mentioned in the document. The challenge here is that the length n of document d may be very long, even exceeding $100k$ tokens. Therefore, directly processing the entire document d may incur high computational cost and inaccurate response results.

LONGAGENT adopts a *divide-and-conquer* strategy, breaking down the document d into $m = \lceil n/l \rceil$ manageable short text chunks and assigning them **one-to-one** to m member agents. The $\lceil \cdot \rceil$ represents the ceiling operator, while l represents the predetermined chunk size (e.g., 1024 or 2048 tokens). l is set to be significantly smaller than the context window size of the member agents (4096 tokens for LLaMA2), thus reserving ample room for subsequent multi-turn interactions. Subsequently, leveraging the characteristic of LLMs to follow instructions, a leader agent and multiple member agents form a team that collaboratively searches these text chunks for clues and reasons to find answers. As illustrated in Figure 2, the collaboration consists of three steps, which we elaborate on in detail in the subsequent sections.

2.2 Collaborative Reasoning

To answer a user's question q, in each interaction round, the leader generates an instruction and broadcasts it to all members. The members read their assigned document chunks and return instruction-relevant cues. If the user's question is a complex multi-hop query, the above leader-member interaction may proceed over multiple rounds. Formally, given an user's question q and the interaction history $S_{i-1} = \{s_1, s_2, ..., s_{i-1}\}$ from the previous $i-1$ rounds, the i^{th} round of interaction commences with a decision-making process by the leader:

$$a_i \sim \text{Leader}(a|S_{i-1}, q), \tag{1}$$

where $a_i \in \{\texttt{QUERY}, \texttt{CONFLICT}, \texttt{ANSWER}\}$ represents the decision outcome of the leader agent in i^{th} round. If $a_i = \texttt{QUERY}$[†], it indicates

[†]CONFLICT and ANSWER actions are described in detail in section 2.3 and 2.4 respectively.

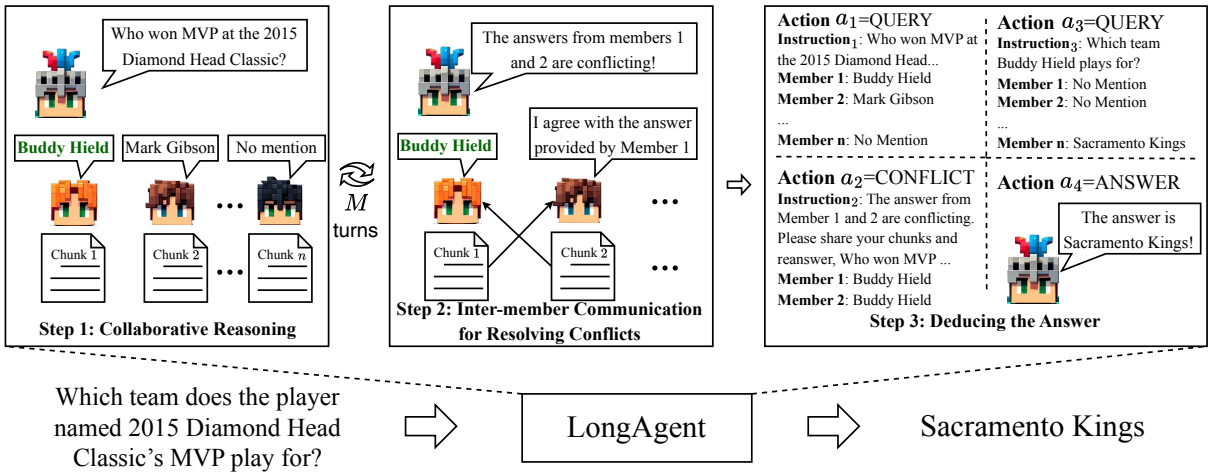

Figure 2: An Overview of LONGAGENT. In Step 1 and Step 2, the leader organizes the team to gather clues from the text chunks and resolve conflicts. After multiple rounds of iteration, the leader reasons out the final answer based on the information accumulated in the interaction history (corresponding to Action a_4 in Step 3).

that the interaction history up to the $(i-1)^{\text{th}}$ round is insufficient to answer the original question q. The leader then initiates the next round of interaction and provides a new instruction. As illustrated in Figure 2, to answer *Which team does the player named 2015 Diamond Head Classic's MVP play for?*, the leader first instructs the members to identify who won the 2015 Diamond Head Classic MVP. Then, based on the cues returned by members (i.e. Buddy Hield), it dynamically generates the next instruction to find *Which team Buddy Hield plays for*. This decomposition of complex queries and multi-round interaction is crucial, as relevant information may be scattered across different chunks of the lengthy document, precluding any single agent from directly answering the original multi-hop question in one interaction round.

2.3 Resolving Conflicts

Due to model hallucinations, members may respond with content not mentioned in their chunks. The interaction in Step 1 of Figure 2 serves as an example, where two members respectively believe *Buddy Hield* and *Mark Gibson* to be the MVP of the 2015 Diamond Head Classic, despite the latter not being mentioned in the text chunk. In such cases, the leader selects the `CONFLICT` action, which invokes the *inter-member interaction* mechanism to resolve the conflict. This mechanism is inspired by the following empirical findings: (a) When a chunk lacks clues relevant to the answer, the member tend to hallucinate responses instead of honestly admitting *No Mention*. (b) However, when the answer is present in the chunk, the member make mistakes less frequently. Therefore, we share text chunks from two members with conflicting responses, expecting the hallucinating agent to revise its response upon receiving the chunk mentioning the correct answer. Formally:

$$\text{Hallucination} = m_i(c_i), \text{Truth} = m_j(c_j), \quad (2)$$

$$\text{Truth} = m_j(c_j \oplus c_i) = m_i(c_j \oplus c_i), \quad (3)$$

c_i and c_j respectively represent two text chunks, where c_j contains the correct answer while c_i does not. m_i and m_j denote two members. $\oplus$ denotes concatenation of two chunks. Our experimental results demonstrate that sharing text chunks is a simple yet effective strategy. The majority of members experiencing hallucination tend to correct their original responses upon receiving the chunk containing the correct answers, resulting in accurate output.

Once the conflicts are resolved, the leader executes the decision process described in Section 2.2 again and selects the next action for the subsequent interaction round.

2.4 Deducing the Answer

When the leader judges that the interaction history contains sufficient information to reason the answer, it executes the action a =`ANSWER` and generates the final answer. Taking the interaction history from Figure 2 (Step 3) as an example, in the first round, the leader performs a_1 =`QUERY` and learns that both *Buddy Hield* and *Mark*

Gibson are potential candidates for the 2015 Diamond Head Classic MVP. Consequently, in the second round, it executes a_2 =CONFLICT and discovers that *Mark Gibson* is a hallucinated answer. Subsequently, in the third round, the leader performs a_3 =QUERY and finds that *Buddy Hield* plays for the Sacramento Kings. At this point, the interaction history contains enough information to derive the answer to the original question, so the leader executes a_4 =ANSWER and outputs *The final answer is Sacramento Kings*.

2.5 How to Construct Agents

LONGAGENT involves both leader agents and member agents, which can be obtained by fine-tuning open-source base models or prompting strong instruction-following models.

Fine-tuning open-source base models. We fine-tune the LLaMA2-7B model to build our agent team. To train the leader agent, we generated $1,000$ interaction trajectories using GPT-4 and manually verified the correctness of these trajectories. Based on this data, the leader agent learns how to decompose complex problems and reason the final answer from interaction history. To train the members to read their assigned text chunks and respond to the leader's instructions, we constructed a $25,000$-sample QA dataset based on the SQuAD dataset (Rajpurkar et al., 2016). Each sample comprises a text chunk and a corresponding question. Among these samples, $10,000$ text chunks contain the answer to the associated question, while the remaining $15,000$ do not mention the answer. For the latter cases, the member is trained to respond with *No Mention*.

Prompting instruction-following models. For aligned models (such as GPT-4, GPT-3.5, etc.), we can use prompting to construct an agent team. For example, a viable prompt for the member agent is *You are an QA expert, adept at searching for relevant information from given documents and providing answers*. Please refer to Appendix D for full prompt templates and interaction trajectories.

3 Experimental Setup

3.1 Evaluation Protocol

Needle-in-a-Haystack PLUS: The *Needle-in-a-Haystack* (Kamradt, 2023) is currently one of the most popular testbed for evaluating LLM's capability to handle long documents. As illustrated in Figure 3, a text (the *needle*) is inserted into a

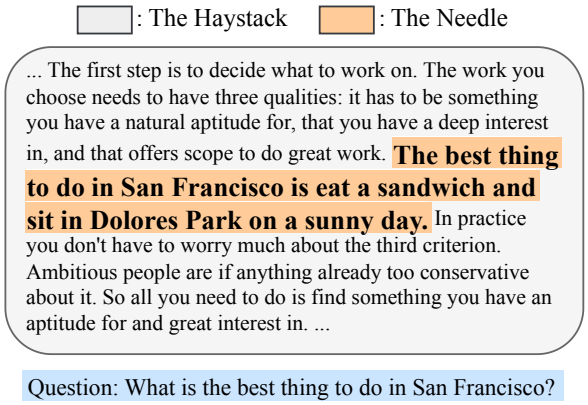

Figure 3: Overview of the *Needle in a Haystack*. By varying the depth percentage α and the haystack length L, we can conveniently construct test samples of different lengths, with critical information situated at varying positions.

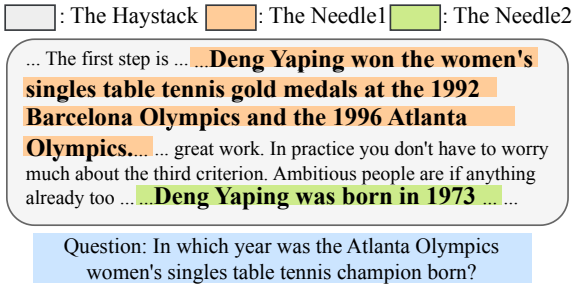

Figure 4: *Multi-needle* setting in our PLUS version.

lengthy document (the *haystack*). The *Needle-in-a-Haystack* evaluates a model's ability to retrieve critical information by testing whether it can accurately answer a given question. In its setup, the **length of haystack** L is defined as the document's token count, while the needle's insertion position is determined by the **depth percentage** α, which represents the percentage of tokens preceding the insertion point relative to the total token count L. By varying L and α, *Needle-in-a-Haystack* comprehensively assesses a model's performance on inputs of different lengths and with critical information positioned at various locations.

Based on this setup, we propose the *Needle-in-a-Haystack PLUS* benchmark. We have made upgrades in three aspects: task difficulty, data diversity, and prevention of data leakage:

(1) Task difficulty: The original *Needle-in-a-Haystack* constitutes a simple QA task, where one needle corresponds to one question and the answer to the question is explicitly stated in the needle. We name this task as *single-needle QA* task. It emphasizes evaluating a model's ability to retrieve

relevant information.

To better assess a model's reasoning capabilities on long documents, we introduce the *multi-needle QA* task, where multiple needles correspond to one question and the answer to the question should refer to more than one needles. For instance, to answer the question in Figure 4, the model must first identify the key entity Deng Yaping from Needle 1, and then infer based on this information that the year mentioned in Needle 2 is the answer to the original question.

(2) Data diversity: In the original *Needle-in-a-Haystack*, for different haystack length L and depth percentages α, only a single needle about San Francisco and its corresponding question (see Figure 3 are provided for evaluation. This could introduce substantial evaluation bias. In contrast, we collect 100 needles from SQuAD dataset for *single-needle* QA and 60 pairs of needles from HotpotQA for *two-needle* QA. Given a haystack length L and a needle position α, we randomly select 10 needles from the 100 needles for single-needle evaluation (select 10 pairs of needles from the 60 pairs of needles for multi-needle evaluation), and report the average performance across 10 runs in our experiments. **It is important to note that there is no overlap between any of the evaluation data and the agent training data described in Section 2.5.**

(3) Prevention of data leakage: LLMs have undergone extensive pre-training and have amassed a wealth of global knowledge. Therefore, the model may directly respond based on the knowledge encoded in its parameters, rather than find the answer from the needle. To more accurately reflect the model's long-document QA capability, we deliberately alter the needle to be unrealistic and expect the model to provide an answer that aligns with the needle, rather than one that conforms to the facts. For example, we may change a needle from "Deng Yaping won the women's singles table tennis gold medals at the 1992 Barcelona Olympics", a real fact, to "Deng Yaping won the women's singles table tennis gold medals at the 2020 Tokyo Olympics", a wrong statement, and expect the model to tell us "2020" for the question "In which year did Deng Yaping win the women's singles table tennis gold medals in the Olympics?".

Other Existing Benchmarks: LongBench (Bai et al., 2023) and InfiniteBench (Zhang et al., 2023) are two widely used long-text evaluation benchmarks currently. We introduce all the QA-related tasks from these two benchmarks for our evaluation. Specifically, LongBench includes 5 long document QA tasks: NarrativeQA, Qasper, Musique, HotpotQA, and 2wikimqa. The document lengths in these tasks range from 0 to $40,000$ tokens. InfiniteBench includes one QA task, which is the Fake Book QA task. The document lengths in this task range from 0 to $200,000$ tokens.

3.2 Evaluation Metrics

For our evaluation, we employ both human evaluation and automatic metrics. Regarding the former, we utilize `accuracy` (Acc) as the evaluation metric. As for the latter, we follow the practice of Bai et al. (2023), adopting the `F1 score` as the metric. For the human evaluation, we recruit three undergraduate students majoring in language-related disciplines from top universities to serve as evaluators. They are tasked with assessing the semantic consistency between the model outputs and the ground truth answers, focusing solely on whether they align factually rather than considering differences in phrasing or other minor details. In Appendix B, we provide a detailed description of how the human annotators were instructed.

3.3 Baselines

PI (Chen et al., 2023b). Extending the context window sizes of RoPE-based pretrained large language models by position interpolation.

Retrieval-Augmented Generation (RAG). For our RAG implementation, we utilize BGE m3 (Chen et al., 2024) as the retriever model and a fine-tuned LLaMA2-7B as the generation model (Same as our member agent). Unless otherwise specified, we set the chunk size to 500 and configure the retriever to return the top-3 text chunks. Additionally, we also provide a strong multi-round retrieval baseline (Trivedi et al., 2023) for evaluation.

Claude2.1 (Anthropic, 2023). The Claude 2.1 released by Anthropic Corporation features a context window of 200K tokens and has significantly reductions in rates of model hallucination.

GPT-4 Turbo (OpenAI, 2023) We use `GPT-4-0125-preview` for all the experiments. The GPT-4 Turbo model from OpenAI offers a context window of 128K and can process text exceeding 300 pages within a single prompt.

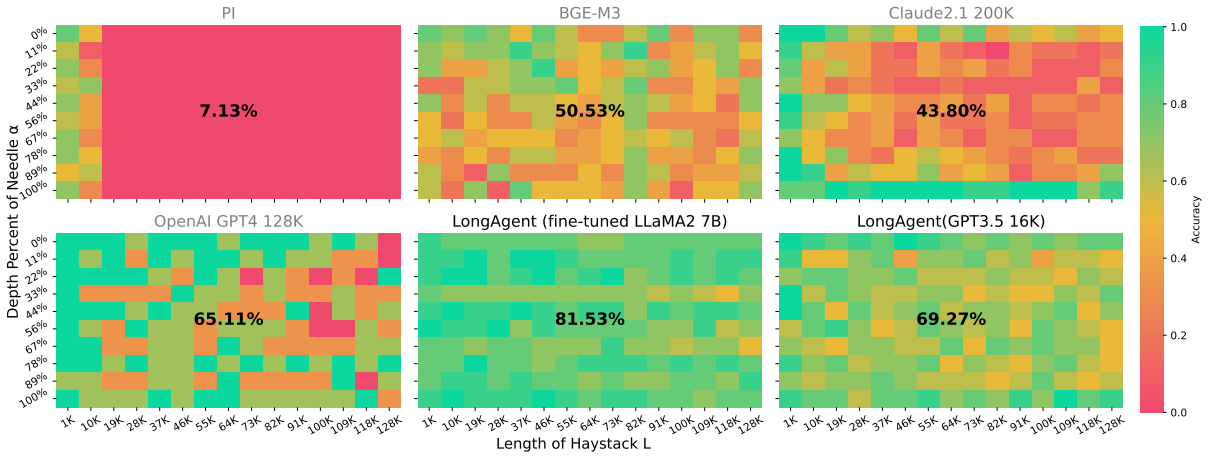

Figure 5: Single-Needle QA Results with *Needle-in-a-Haystack PLUS*. With the help of LONGAGENT, LLaMA2-7B achieves an average accuracy improvement of 16.42% compared to GPT-4 across the range from 1k to 128k (increasing from 65.11% to 81.53%). The bold black numbers in each subfigure indicate the average accuracy.

In Figures 10 and 11 in the Appendix, we additionally provide more baselines for comparison, including Yarn (Peng et al., 2023), and the RAG methods based on TF-IDF and BM25 retrievers.

4 Results and Discussion

4.1 Overall Performance

To demonstrate the superiority of LONGAGENT in handling long texts, we compare it against powerful commercial models GPT-4 Turbo and Claude 2.1, as well as the most popular academic methods for long-text processing, PI and RAG (BGE M3).

Through multi-agent collaboration, LLaMA with a 4k-long context window is able to handle QA of up to 128k-token-long documents.

We fine-tune LLaMA2-7B using the method described in Section 2.5 to construct the leader and member agents. The results for *Single-Needle QA* and *Multi-Needle QA* in the *Needle-in-a-Haystack PLUS* are shown in Figure 5 and 6, respectively. Comparing the subfigures titled `OpenAI GPT4 128k` and `LongAgent (fine-tuned LLaMA2 7B)`, we find that LONGAGENT outperforms GPT-4 across haystack length ranging from 1k to 128k, with an average improvement of 16.42% and 1.63% on Single-Needle QA and Multi-Needle QA respectively.

Additionally, we fine-tune PI and YaRN using QA data of lengths 1k to 16k (training on longer contexts led to OOM errors on our GPUs). From the subplots titled `PI` in Figures 5 and 6, and the subplots titled `YaRN` in Figures 10 and 11 in the appendix, we can see that these methods fail to effectively extrapolate to the 19k-128k input range, while LONGAGENT's accuracy does not degrade within the 128k range as input length increases, demonstrating its advantage in extrapolation ability.

Tables 1 and 2 present a detailed comparison between LONGAGENT and RAG baselines, including a strong multi-round retrieval method. LONGAGENT demonstrates clear advantages regardless of changes in the number of documents returned by the retriever or increases in retrieval rounds. This outcome is unsurprising, as RAG efficiently recalls key text chunks through simple retrievers, which lacks strong fine-grained semantic modeling capabilities. LONGAGENT, utilizing multiple Member Agents to directly read all text chunks, exhibits stronger semantic comprehension compared to retrievers. Overall, considering efficiency, RAG methods are better suited for larger corpora. However, as sequence lengths decrease to several hundred thousand tokens, LongAgent's superiority in semantic understanding becomes increasingly appealing.

For aligned LLMs like GPT-3.5, LONGAGENT can work without fine-tuning.

As described in Section 2.5, we prompted GPT-3.5 to play the roles of the leader and member agents for long document QA. Although GPT-3.5 only has a 16k context window, LONGAGENT allows it to handle inputs far exceeding 16k tokens length out-of-the-box. Comparing the subfigure titled `OpenAI GPT4 128k` and `LongAgent (GPT3.5 16k)` in Figure 5, we find that GPT-3.5 achieves a 6.78% accuracy improvement on Single-

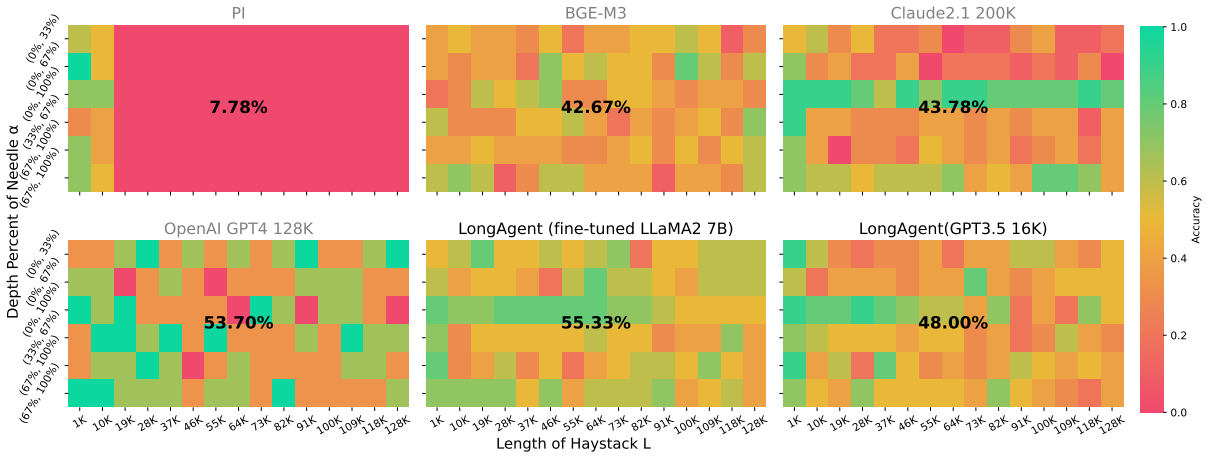

Figure 6: Multi-Needle QA Results with *Needle-in-a-Haystack PLUS*. With the help of LONGAGENT, LLaMA2-7B model achieves an average accuracy improvement of 1.63% compared to GPT-4 across the range from $1k$ to $128k$ (increasing from 53.70% to 55.33%). The two percentage values on the y-axis represent the depth percentages of Needle 1 and Needle 2 respectively. The bold black numbers in each subfigure indicate the average accuracy.

Benchmark	**Tasks**	RAG-Multi-Round (GPT3.5 16k)		GPT-4		LONGAGENT (GPT3.5 16k)	
		ACC.	F_1	ACC.	F_1	ACC.	F_1
LongBench	NarrativeQA	0.560	0.2097	0.600	**0.2785**	**0.680**	0.2453
	Qasper	0.520	0.1839	0.560	**0.2210**	**0.620**	0.1968
	MuSiQue	0.380	0.2255	**0.460**	**0.3924**	0.400	0.2815
	HotpotQA	0.460	0.3527	**0.540**	**0.5613**	0.520	0.4836
	2WikiMQA	0.540	0.4103	0.540	**0.4661**	**0.560**	0.4415
InfiniteBench	FakeBookQA	0.400	0.1362	0.500	0.2144	**0.520**	**0.2258**

Table 1: Performance comparisons on more long-document QA tasks. Considering evaluation cost of GPT-4, we randomly selected 50 samples per task for evaluation. **Bold text** and underlined text respectively indicate the first and second-ranked results. Through LONGAGENT, GPT-3.5 is able to achieve long-text QA performance similar to, or even surpassing, that of GPT-4. This result is non-trivial, as GPT-3.5's reasoning capability is weaker than GPT-4's, and its context window (16k) is much smaller than GPT-4's (128k)

Tasks	Single-Needle		Multi-Needle	
	ACC.	F_1	ACC.	F_1
RAG-Top1	0.4694	0.6866	0.4783	0.3923
RAG-Top3	0.5053	0.6973	0.4267	0.3722
RAG-Top5	0.5522	0.7175	0.4589	0.3827
RAG-Top3-Multi-Round	0.5419	0.7102	0.4943	0.4212
LONGAGENT(LLaMA7B)	**0.8153**	**0.8277**	**0.5533**	**0.5291**

Table 2: A detailed comparison of LONGAGENT and RAG methods in *Needle-in-a-Haystack-PLUS* benchmark. RAG-Topk refers to the retriever returning k relevant text chunks. *Multi-Round* indicates performing multiple rounds of retrieval.

Needle QA. In the multi-needle setting shown in Figure 6, it also matches GPT-4's performance. In Table 1, we also test the QA-related tasks in LongBench and InfiniteBench. The results show that LONGAGENT(GPT3.5-16k) outperforms GPT-4 on more than half of the tasks. Given that the original context window of GPT-3.5 is 16k, far smaller than the 128k of GPT-4, and that the capability of GPT-3.5 itself is weaker than GPT-4, the above results are sufficient to demonstrate the effectiveness of LONGAGENT.

4.2 Hallucination Analysis

We find that LONGAGENT's errors mainly stem from a specific type of member hallucination: when the assigned text chunk does not contain information relevant to the leader's query, the member sometimes fabricates a response. This section investigates two key factors, the recipes in the training data and the document length (i.e., the length of the text chunk assigned to the

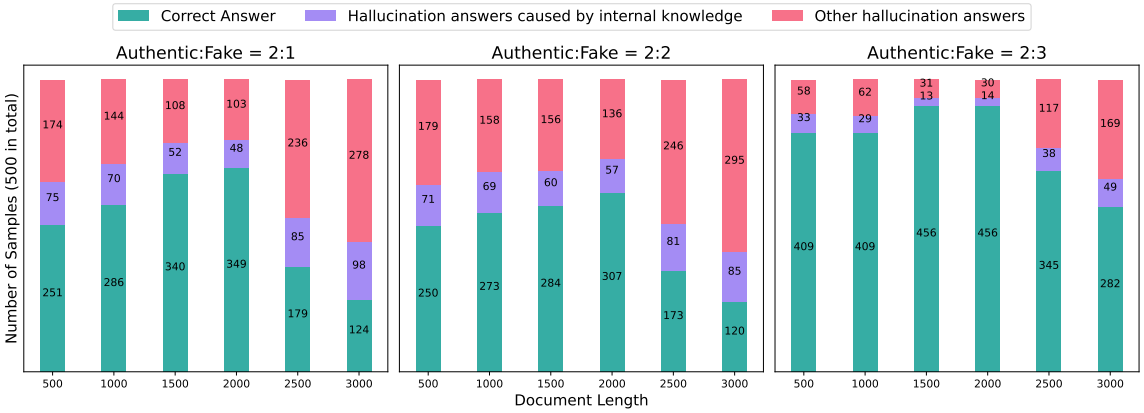

Figure 7: The influence of training data recipe on model hallucinations. Each piece of data is a 3k-token-long document paired with a question. The data is considered *authentic* if the question can be answered using the document. Conversely, it is deemed *fake* if the document lacks content relevant to the question. We employ various data recipes to train the model, adjusting the ratio of authentic to fake data (2:1, 2:2, and 2:3). We assess the model's tendency to produce hallucinated responses by using fake data, with document lengths varying between 500 and 3000 tokens. For each specific document length, we evaluate using 500 pieces of fake data. Ideally, the model should consistently respond with *No Mention* to all 500 items. However, it may also generate hallucinated answers based on its internal knowledge or other factors.

member) on member hallucination. Comparing the three subfigure in Figure 7, as the proportion of *fake* type data in the member's training data increases (from $2:1$ to $2:3$), the percentage of correctly answering *No Mention* significantly improves. Particularly when *authentic*:*fake* reaches $2:3$, in the 4 test document length groups from 500 to $2,000$, the number of correctly answered documents exceeds 400 in each group.

Meanwhile, the length of the input document is also an important factor affecting member hallucination. The three subfigure in Figure 7 all show the following trend: as the document length increases from 500 to $2,000$ tokens, the number of samples receiving correct responses gradually increases. This is mainly because the member's training data consists of $3,000$-token-long documents. The increase in test document length gradually reduces the length gap between training and test data. However, when the test document length exceeds $2,000$, the member's degree of hallucination starts to increase. We speculate that this may be because the input document length is gradually approaching LLaMA2's pretraining length (4k), and LLaMA2's own sequence modeling capability becomes increasingly inadequate at such lengths.

4.3 Ablation Study

In this section, we verify the effectiveness of the cross-member communication mechanism in mitigating member hallucination. As shown in

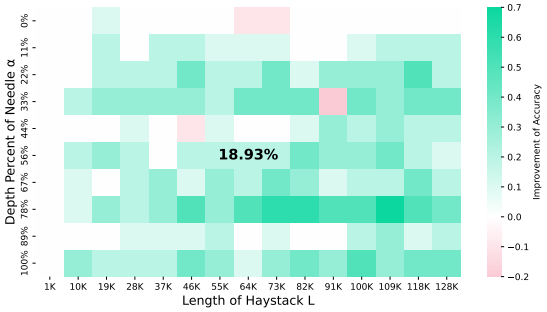

Figure 8: Improved accuracy through *inter-member communication* mechanism. '18.93%' denotes the average Acc improvement across different α and L.

Figure 8, after introducing the *cross-member communication mechanism*, almost all the cells are green. This means that the *cross-member communication mechanism* achieved positive accuracy improvements under different settings of haystack length and needle depth percentage (18.9% accuracy improvement on average). Furthermore, the number of members increases with the length of the text, and the number of members experiencing hallucinations also grows. In this context, the improvement in accuracy brought about by conflict resolution becomes even more evident.

4.4 Efficiency Advantage

Thanks to chunking of long texts, LONGAGENT's time complexity for processing long texts is $\mathcal{O}(N)$. In this subsection, we empirically verify this

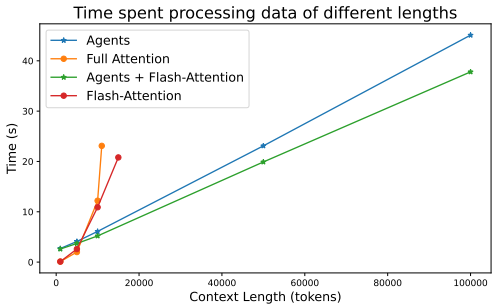

Figure 9: LONGAGENT scheme exhibits significantly superior time and memory efficiency compared to directly perform full attention on long texts.

point. As shown in Figure 9, the latency of LONGAGENT within the range of $1k$-$100k$ almost grows linearly with length. For Full Attention, which has quadratic complexity, the inference latency increases rapidly regardless of the use of techniques such as flash attention. The latency of Full Attention when processing 10k tokens has already exceeded that of LONGAGENT processing $50k$ tokens. Furthermore, without specific memory optimization techniques, a single A100 GPU with 80G memory can only support text inference up to 11k in length, and even with flash attention, this number can only be increased to $15k$. Under the same settings, LONGAGENT can process contexts of around $100k$ with less than 40G of memory.

5 Related Works

Several methods have been proposed to extend the positional encoding (PE) for handling longer sequences. Initially, approaches like RoPE and PI (Chen et al., 2023b) attempted to interpolate position indices within pre-trained limits, but neglected frequency variations. Recent advancements include "NTK-aware" (Bloc97, 2023a) interpolation and "Dynamic NTK" (Bloc97, 2023b) interpolation, which address high-frequency component losses. Additionally, "NTK-by-parts" (Bloc97, 2023c) interpolation outperforms others when fine-tuned on longer-context data. Another popular approach for managing longer sequences involves constraining global causal attention to local attention. ReRoPE (Su, 2023) truncates context lengths during pretraining and LM-Infinite (Han et al., 2023) restricts attention to a chevron-shaped window. Mohtashami and Jaggi (2023) insert landmark tokens after text fragments, while Zhang et al. (2024) propose beacon tokens for summarizing fragments. In contrast, our method effectively circumvents the risk of losing valuable contextual information while utilizing only a small amount (hundreds of agent interaction tracks) for fine-tuning, thereby reducing training costs.

6 Conclusions

This paper proposes LONGAGENT, a novel long-document QA approach based on multi-agent collaboration. The proposed *inter-member communication* mechanism alleviates the member hallucination when they reading documents, thus facilitating effective management by the leader of dozens to hundreds of members. We have also developed *Needle-in-a-Haystack PLUS* to facilitate a comprehensive assessment of the LLM's understanding on long documents. Our experimental results indicate that LONGAGENT offers a promising alternative for long-text QA.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No.62076069,62206057,61976056), Shanghai Rising-Star Program (23QA1400200), and Natural Science Foundation of Shanghai (23ZR1403500), Program of Shanghai Academic Research Leader under grant 22XD1401100.

Limitations

The direction of long text processing based on multi-agent collaboration still has many interesting points yet to be explored: (1) More diverse types of agents and broader task scopes: Current agents already possess capabilities such as tool invocation, coding, and multimodal understanding. Therefore, an intriguing question is how to fully leverage these capabilities of agents and handle long sequences of document summarization, repository-level code processing, video understanding, and so on. (2) Further alleviating hallucinations: Reducing hallucinations during the multi-agent collaboration process is crucial for the ultimate collaborative effect. Although this paper proposes a mechanism to mitigate hallucinations, further research on this issue is still highly necessary.

References

Anthropic. 2023. Model card and evaluations for claude models. Website. https://www.anthropic.com/product.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023. Longbench: A bilingual, multitask benchmark for long context understanding.

2023a Bloc97. 2023a. Ntk-aware scaled rope allows llama models to have extended (8k+) context size without any fine-tuning and minimal perplexity degradation. https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/.

2023b Bloc97. 2023b. Dynamically scaled rope further increases performance of long context llama with zero fine-tuning. https://https://www.reddit.com/r/LocalLLaMA/comments/14mrgpr/dynamically_scaled_rope_further_increases//.

2023c Bloc97. 2023c. Ntk-aware interpolation "by parts" correction, 2023. URL https://github.com/jquesnelle/scaled-rope/pull/1.

Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Bing. 2023a. Clex: Continuous length extrapolation for large language models.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023b. Extending context window of large language models via positional interpolation.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2023c. Longlora: Efficient fine-tuning of long-context large language models.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models?

Hongye Jin, Xiaotian Han, Jingfeng Yang, Zhimeng Jiang, Zirui Liu, Chia-Yuan Chang, Huiyuan Chen, and Xia Hu. 2024. Llm maybe longlm: Self-extend llm context window without tuning.

Greg Kamradt. 2023. Needle in a haystack - pressure testing llms. Website. https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts.

Amirkeivan Mohtashami and Martin Jaggi. 2023. Landmark attention: Random-access infinite context length for transformers.

OpenAI. 2023. Gpt-4 technical report.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models.

Ofir Press, Noah A. Smith, and Mike Lewis. 2022. Train short, test long: Attention with linear biases enables input length extrapolation.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Jianlin Su. 2023. Rectified rotary position embeddings. https://github.com/bojone/rerope.

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 the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Szymon Tworkowski, Konrad Staniszewski, Mikołaj Pacek, Yuhuai Wu, Henryk Michalewski, and Piotr Miłoś. 2023. Focused transformer: Contrastive training for context scaling.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. 2024. Soaring from 4k to 400k: Extending llm's context with activation beacon.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Qihao Wu, Junhao Chen, Zihang Xu, Zhenning Dai, Xu Han, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2023. Infinitebench: 128k long-context benchmark for language models.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Peng Cui, Tiannan Wang, Zhenxin Xiao, Yifan Hou, Ryan Cotterell, and Mrinmaya Sachan. 2023. Recurrentgpt: Interactive generation of (arbitrarily) long text.

Appendix

A The conceptual differences between LongAgent, Long-context LLM, and the Multi-round retrieval system

Conceptually, LongAgent is neither a long-context LLM nor a traditional multi-round retrieval system.

(1) It differs from retrieval systems in several key aspects: Typical RAG methods use retrievers designed for internet-scale corpora, balancing retrieval efficiency and precision. These efficient retrievers can only roughly assess semantic relevance between text chunks and queries. In contrast, LongAgent isn't designed for such large-scale corpora. It aims to process given long inputs (128k length texts, not large-scale corpora) far exceeding a single model's context window. Instead of prioritizing efficiency, LongAgent focuses on fine-grained semantic understanding of given long inputs, which is a strength of LLMs/agents.

In the processing pipeline, LongAgent can be viewed as a refined processing stage following the RAG retriever's initial document recall. Considering that when the number of recalled documents is small, the retriever's coarse-grained recall might miss crucial text chunks. With LongAgent, a single LLM can process input text far exceeding the context window size, allowing the retriever to return more relevant chunks (e.g., from top 5 to top 500), effectively preventing the loss of key information.

(2) LongAgent differs from long-context LLMs in that it's a agent-collaboration method built on top of LLMs, not a specific model. It can be seen as a plug-and-play LLM capability expansion solution, enabling LLMs to process long inputs beyond their context window. Moreover, as the capabilities of LLM backbones improve, the overall performance of multi-model collaboration will also enhance. Therefore, we believe that research on multi-model collaboration is a promising direction in addition to expanding the context window size of a single model.

B Human Evaluation Instruction

In table 3 and 4, we offer the instructions for human evaluation. When calculating accuracy, if a sample receives a score of 0 from a human evaluator, we consider it incorrect. If the human evaluator gives a score of 1 or 2, we consider the sample to be correct.

C Additional Results

Figures 10 and 11 provide additional experimental results of more baselines on the large document retrieval benchmark.

D Prompt Used in Multi-agent Collaboration

Table 5-7 presents the prompt templates used in the multi-agent collaboration process.

Dear Evaluator:
Thank you for participating in this human evaluation of the QA model. Your expertise is crucial for this assessment. Please read the following guidelines carefully to ensure the accuracy and consistency of the evaluation.

1. Evaluation Purpose: This evaluation aims to determine the semantic consistency between the model output and the ground truth. We primarily focus on factual consistency, judging whether key facts, data, concepts, and other core information match, rather than differences in expression or other details.

2. Evaluation Process:

2.1 You will receive a series of questions, model answers, and corresponding ground truths.

2.2 Carefully read the question, model answer, and ground truth.

2.3 Evaluate the semantic and factual consistency between the model answer and the ground truth.

2.4 Score using the provided scoring criteria.

2.5 Provide a brief explanation for each score, explaining your reasoning.

3. Scoring Criteria: Please use the following three-level scoring criteria:

2 points: Completely consistent The model answer fully matches the ground truth in facts and semantics. All key information is correct and complete.

1 points: Partially consistent The model answer roughly matches the ground truth in facts and semantics. Core information is generally correct, but there are some ambiguities/inaccuracies.

0 points: Inconsistent The model answer has obvious contradictions or serious errors in key facts compared to the ground truth. Most or all core information is incorrect.

4. Evaluation Considerations:

4.1 Focus on factual content: Don't overly focus on differences in expression, wording, or style. Primarily assess the accuracy of core information.

4.2 Remain objective: Make judgments based on facts, avoiding personal preferences influencing evaluation results.

4.3 Consider context: Some expressions may have the same semantics in specific contexts; please consider the background of the question.

4.4 Ignore minor differences: If differences don't affect core meanings and key facts, they can be considered consistent.

4.5 Consistency: Try to maintain consistency in scoring standards; you can review previous scores as a reference.

Table 3: Human evaluation instructions part 1. Due to page space limitations, the instruction is divided into two parts. For the second part of the table, please refer to Table 4.

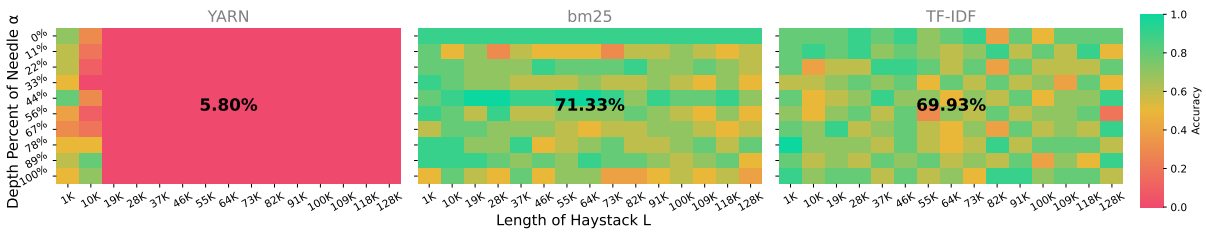

Figure 10: Addition Results of Needle-in-a-Haystack PLUS in Single-Needle QA Setting

5. Evaluation Examples Question:
Question: "Who invented the telephone?"
Ground Truth Answer: "Alexander Graham Bell invented the telephone. He successfully transmitted human voice through a telephone on March 10, 1876."
Example Model Answers and Scores:

Answer 1: "The telephone was invented by Alexander Graham Bell in the 1870s."
Score: 2 points (Completely consistent)
Reason: Although no specific date is provided, the core facts (inventor and approximate time) are correct and consistent.
Answer 2: "Alexander Graham Bell invented the telephone in the 19th century, an invention that revolutionized human communication."
Score: 1 point (Partially consistent)
Reason: The inventor is correct, the time range is broad but not incorrect. No specific invention date is provided, but the additional information doesn't affect the core facts.
Answer 3: "The telephone was invented by Thomas Edison in 1876."
Score: 0 points (Inconsistent)
Reason: The inventor information is incorrect. Although the year is correct, it doesn't compensate for the error in the key fact.

6. Quality Control

6.1 Some of your evaluation results will be compared with those of other evaluators to ensure consistency.

6.2 If you encounter situations that are difficult to evaluate, please record them and discuss with the evaluation team after the assessment.

7. Schedule Please complete the evaluation work by the specified date. It is recommended that daily evaluation time does not exceed 4 hours to ensure evaluation quality.
If you have any questions or suggestions about the evaluation process, please feel free to contact the evaluation team. Your professional assessment will make an important contribution to improving the performance of the QA model.
Thank you again for your participation.

Table 4: Human evaluation instructions part 2. Due to page space limitations, the instruction is divided into two parts. For the first part of the table, please refer to Table 3.

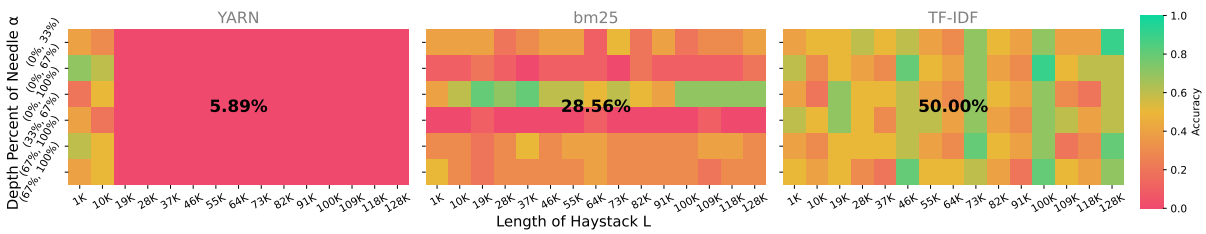

Figure 11: Addition Results of Needle-in-a-Haystack PLUS in Multi-Needle QA Setting

You are the leader of a team of {member_nums} members. Your team will need to collaborate to solve a task. The rule is:
1. Only you know the task description and task objective; the other members do not.
2. But they will receive different documents that may contain answers, and you need to send them an instruction to query their document.
3. Your instruction need to include your understanding of the task and what you need them to focus on. If necessary, your instructions can explicitly include the task objective.
4. Finally, you need to complete the task based on the query results they return.

 # Task Description:
Answer the question based on the given passages. The answer must be extracted from the given passages.

 # Task Objective:
{user_question}

 # Generate Instruction for Members:
Now, you need to generate an instruction for all team members. You can ask them to answer a certain question, or to extract information related to the task, based on their respective documents.
Your output must following the JSON format: {{"type": "instruction", "content": "your_instruction_content"}}

Table 5: Prompt templates for the Leader to understand user queries and generate instructions to members

 # Document:
{member_chunk}

 # Instruction:
Answer the question based on the given passages. The answer must be extracted from the given passages.
Question: {Leader_Instruction}

You are an experienced reader; please summarize the content in the document related to the instructions in a <scratchpad> tag, then describe your response." Your output must following the JSON format: {{"type": "response", "content": "your_response_content"}}
The "content" needs to be as concise as possible.

Table 6: Prompt template for Member to read text chunks and respond to Leader instructions

Here are the responses from all the members. Each member sees different segments of a document, and these segments do not intersect with each other. The correct answer may appear in any one or several members' responses.
Note that if a minority of members find information relevant to the question while the majority reply that the document does not contain information relevant to the question, you should pay attention to the replies from those members who found relevant information.
 # Member Response:
`{Member_Response}`

 # Task Description:
Answer the question based on the given passages. The answer must be extracted from the given passages.

 # Task Objective:
`{User_Question}`

 # Determination:
Based on the above information, you need to determine if you can solve the task objective. You have two choices:
1. If members' responses cannot solve the task objective, or if their responses contain conflicting answers, provide a new instruction for them to answer again.
2. Else, if the task objective can be solved, give your final answer as concisely as you can, using a single phrase if possible. Do not provide any explanation.
Your output must following the JSON format: {{"type": "answer", "content": "your_answer_content"}} or {{"type": "instruction", "content": "your_instruction_content"]}}

Table 7: Prompt template for Leader to make decisions and generate new instructions.