LLM×MapReduce: Simplified Long-Sequence Processing using Large Language Models

Zihan Zhou[1,3*], **Chong Li**[4*], **Xinyi Chen**[5*], **Shuo Wang**[2†], **Yu Chao**[2],
Zhili Li[6], **Haoyu Wang**[6], **Qi Shi**[2], **Zhixing Tan**[2],
Xu Han[2], **Xiaodong Shi**[1,3†], **Zhiyuan Liu**[2], **Maosong Sun**[2]
[1]School of Informatics, Xiamen University [2]Tsinghua University
[3]Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan (Xiamen University), Ministry of Culture and Tourism
[4]Peking University [5]Nankai University [6]BUPT

Abstract

We propose a training-free framework that enables large language models (LLMs) to effectively process long texts, using a divide-and-conquer strategy for comprehensive document understanding. The proposed LLM×MapReduce framework splits the entire document into several chunks for LLMs to read and then aggregates the intermediate outputs to produce the final response. The main challenge for divide-and-conquer long text processing frameworks lies in the risk of losing essential long-range information due to document splitting, which can lead the model to produce incomplete or incorrect answers based on the segmented texts. Disrupted long-range information can be classified into two categories: inter-chunk dependency and inter-chunk conflict. We design a *structured information protocol* to better cope with inter-chunk dependency and an *in-context confidence calibration* mechanism to resolve inter-chunk conflicts. Experiments demonstrate that LLM×MapReduce outperforms representative open-source and commercial long-context LLMs and is compatible with several models. Our framework can also function as a data synthesis engine, capable of generating high-quality long-alignment data using only short-context LLMs. [1]

1 Introduction

Large language models (LLMs) exhibit impressive performance across a wide range of complex tasks (OpenAI, 2023), including question answering (Anthropic, 2023), code generation (Luo et al., 2024), and solving mathematical problems (Luo et al., 2023). However, due to their quadratic computational complexity and a lack of high-quality long training examples, most LLMs are trained with a limited window size (Touvron et al., 2023a,b; Jiang et al., 2023). This context limit restricts the application of modern LLMs to long-sequence processing tasks. In response to this issue, several researchers have focused on extending the context length of LLMs. Existing studies can be broadly categorized into two types: training-based and training-free methods.

For training-based extension methods, it is necessary to prepare long training data and allocate substantial computational resources to support the additional training (Xiong et al., 2023; Chen et al., 2024). Although these training-based methods can effectively extend the context length of LLMs, they may be inapplicable in scenarios where sufficient computational resources and high-quality long texts are unavailable.

In contrast, training-free context extension approaches aim to overcome the length limitations of LLMs without modifying their parameters (Xiao et al., 2024b,a). A prominent approach within this field is the divide-and-conquer strategy, which processes long sequences by breaking them into shorter, more manageable chunks (Wang et al., 2024; Zhao et al., 2024; Zhang et al., 2024b; Qian et al., 2024). LangChain (Chase, 2022) initially introduces the MapReduce method, where text segments are processed in parallel during the map stage, followed by the aggregation of intermediate results across all segments to predict the final output. The major challenge for this kind of method is that different segments are processed independently, which may break some essential long-range information. Disrupted long-range information can be divided into two categories: (1) **inter-chunk dependency**, where evidence is spread across different chunks and relies on each other; and (2) **inter-chunk conflict**, where evidence across chunks is contradictory, requiring the model to resolve these conflicts in order to predict the final answer.

In this paper, we introduce LLM×MapReduce, a training-free framework for processing long texts

[*]Equal Contribution.
[†]Correspondence to Shuo Wang and Xiaodong Shi.
[1]The code is publicly available at https://github.com/thunlp/LLMxMapReduce.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27664–27678

that employs a divide-and-conquer approach, enabling models with short context windows to effectively handle long contexts. To address the challenges of inter-chunk dependency and conflict, we introduce a **structured information protocol** and an **in-context confidence calibration** mechanism. The structured information protocol defines the information passed from the map stage to the reduce stage, ensuring the model has the critical inputs needed to infer the correct answer when aggregating different chunks. In-context confidence calibration allows the model to assign a reliable confidence score to the output of each chunk, aiding in effectively resolving inter-chunk conflicts. We evaluate the proposed method on various long-text benchmarks, and the experimental results show that our approach outperforms both closed- and open-source LLMs in terms of both performance and efficiency. Through ablation experiments, we further validate the effectiveness of each component in LLM×MapReduce, reaffirming the importance of explicitly addressing cross-chunk dependencies and conflicts.

Moreover, the proposed framework can also serve as a data generation engine, leveraging short-context LLMs to synthesize long-form alignment data, thereby achieving the goal of "letting short teach long." Annotating long-range texts is time-consuming and labor-intensive, making data synthesis an essential avenue for constructing long-form alignment datasets. By applying a divide-and-conquer approach, we aggregate information from a long document layer by layer, constructing pyramid-shaped textual representations. This pyramid structure enables explicit control over the amount of information when generating QA pairs for the corresponding document. As a result, compared to directly using the entire document (Bai et al., 2024) or focusing on a specific local section (An et al., 2024), our method generates QA pairs that cover varying amounts of information, offering a range of difficulty levels. Experimental results demonstrate that the resulting dataset, Pyramid-Align, enables better model performance than LongAlign, a representative long-range alignment dataset. Finally, we fine-tune an 8B model using Pyramid-Align and conduct inference with LLM×MapReduce, achieving performance better than GPT-4, thus demonstrating the effectiveness of our framework.

Our main contributions include:

- We propose a divide-and-conquer framework for long-sequence processing that explicitly tackles cross-chunk dependencies and conflicts through a structured information protocol and in-context confidence calibration.
- We evaluate the performance and efficiency of the proposed framework against several representative baselines. The results highlight the superiority of our approach, which is also compatible with various LLMs.
- We extend the framework into a data synthesis engine that uses short-context LLMs to generate long-range alignment data. The resulting Pyramid-Align dataset enables an 8B model to outperform GPT-4 on long-sequence tasks.

2 Related Works

Divide-and-Conquer Long-Sequence Processing Thanks to the flexibility and scalability of divide-and-conquer methods for processing long sequences, many researchers have explored using this approach to extend the effective context length of existing LLMs. LangChain (Chase, 2022) is a pioneering framework that breaks long documents into smaller chunks for LLMs to process. Similarly, in XL^3M (Wang et al., 2024), long texts are divided into multiple short sub-contexts, each paired with a question. Relevant segments are then selected using LLMs and combined chronologically to generate the final answer. Segment+ (Shi et al., 2024) also splits long texts using structured notes and a filtering module to manage information flow. Recently, LongAgent (Zhao et al., 2024) introduces a multi-agent framework consisting of a leader agent and several member agents, each responsible for processing a chunk. The leader agent organizes the member agents into groups and then randomly selects one member from each group to aggregate the final answer. Our experiments show that LongAgent's aggregation mechanism does not effectively address inter-chunk dependencies and conflicts, as the random selection of members can lead to the loss of important evidence. Unlike LongAgent, which processes multiple chunks in parallel, Chain-of-Agents (CoA) (Zhang et al., 2024b) processes split chunks sequentially using an accumulated summary. However, since CoA's workflow does not explicitly address inter-chunk conflicts, key clues in the memory may be overwritten when processing subsequent chunks. LC-Boost (Qian et al.,

2024) defines an action space and selects appropriate actions for sequentially processing chunks. To address inter-chunk conflicts, LC-Boost adaptively either appends new evidence or updates the summary. However, in complex cases where historical and current information conflict, LC-Boost may struggle to fully resolve the issue when relying solely on the accumulated summary and the current text. Augmented with the structured information protocol and in-context confidence calibration, our proposed LLM×MapReduce framework can better cope with the inter-chunk dependencies and conflicts.

Long-Range Alignment Datasets Alignment is a crucial step in training effective LLMs. Due to the prohibitively high cost of having human experts create QA pairs for long document understanding, several studies have proposed automatic data synthesis methods for constructing long-alignment datasets (Chen et al., 2024; Bai et al., 2024; An et al., 2024). LongAlign (Bai et al., 2024) leverages an existing long-context LLM (i.e., Claude-2.1) to generate QA pairs from entire long documents. In contrast, An et al. (2024) propose using local chunks to generate QA pairs, which are then concatenated into long documents. Rather than relying solely on global or local information, we propose structuring the document as a pyramid, with different levels of nodes used to generate QA pairs. The resulting Pyramid-Align dataset contains questions that cover varying amounts of information, offering more diverse supervision.

3 Approach

3.1 Problem Description

In real-world scenarios, users may require the LLM to comprehend one or more lengthy documents that far exceed the model's effective context window. Formally, let X represent the user-provided long text and L denote the model's effective context length. In this work, we focus on cases where $|X| \gg L$, where $|X|$ represents the length of X. we partition the input text X into a series of chunks $\{x_1, x_2, \cdots, x_n\}$. The chunking process considers both token length and natural semantic boundaries, such as paragraphs. Our primary aim is to fit entire paragraphs within a chunk while maximizing the chunk size, ensuring each chunk x_i is within the model's context length L. If a paragraph itself exceeds L, it is further subdivided based on punctuation to maintain sentence integrity as much as possible. For a given user query Q, the LLM, parameterized by $\boldsymbol{\theta}$, processes each chunk to generate intermediate outputs, which are then aggregated to predict the final answer.

3.2 Workflow of Our Framework

Figure 1 depicts the overall framework of the proposed LLM×MapReduce framework. Like LangChain (Chase, 2022), the LLM×MapReduce workflow consists of three stages: map, collapse, and reduce. During the map stage, we utilize an LLM as the map model to extract the necessary information for each chunk x_i:

$$s_i = f_{\text{map}}\left(x_i, Q; \boldsymbol{\theta}\right), \tag{1}$$

where Q is the user query and f_{map} represents the map function powered by the LLM, parameterized by $\boldsymbol{\theta}$. Our experiments show that the design of the mapped results, $\{s_1, \cdots, s_N\}$, is crucial for enabling the divide-and-conquer framework to effectively comprehend long documents. In this work, we propose a structured information protocol aimed at improving communication efficiency between the different stages.

In some cases, the input text is extremely long, resulting in mapped results that still exceed the context window of the LLM being used. To mitigate this issue, we introduce a collapse stage that compresses the structured outputs generated during the map stage in a manner similar to the MapReduce approach employed in LangChain (Chase, 2022). We divide the N mapped results into K groups, forming each group g_j by sequentially concatenating results until the next addition would exceed the context limit L. For the j-th group of mapped results g_j, the collapsed result can be given by

$$c_j = f_{\text{collapse}}\left(g_j, Q; \boldsymbol{\theta}\right). \tag{2}$$

It is important to note that the structure of each collapsed result c_j remains the same as that of each mapped result s_i. If the total length of the mapped results $\{s_1, \cdots, s_N\}$ is less than L, we use the mapped results directly as the collapsed results for the reduce stage. If the collapsed results $\{c_1, \cdots, c_K\}$ still exceed L, we iteratively apply the collapse function f_{collapse} until their length is reduced to less than L. Briefly, we use $\{c_1, \cdots, c_K\}$ to denote the final output of the collapse stage.

Finally, in the reduce stage, the final response is generated based on the collapsed results:

$$a = f_{\text{reduce}}\left(\{c_1, \cdots, c_K\}, Q; \boldsymbol{\theta}\right). \tag{3}$$

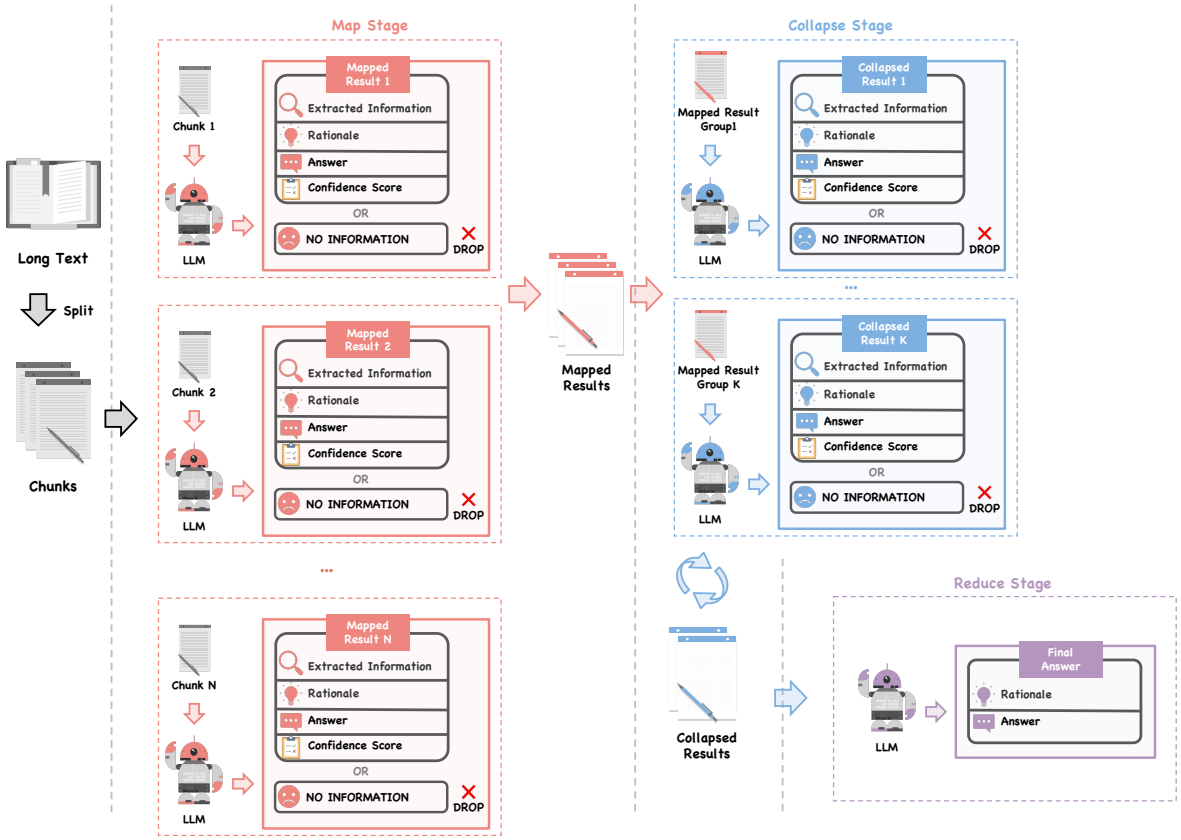

Figure 1: Overview of the proposed LLM×MapReduce framework. After dividing the provided long text into a series of chunks, the model processes each chunk to extract an information structure containing the essential content needed to address the query. This is referred to as the map stage in our framework. The mapped results are then compressed during the collapse stage, preparing them for the reduce stage. The collapse stage ensures that the input to the reducing model remains within its effective length (i.e., L). Based on the structured outputs from the first two stages (i.e., the map and collapse stages), the reduce model aggregates information from all chunks, resolves inter-chunk conflicts using calibrated confidence scores, and predicts the final answer.

In LLM×MapReduce, we do not need to tune the model parameters $\boldsymbol{\theta}$. Instead, the three functions (i.e., f_{map}, f_{collapse}, and f_{reduce}) are implemented using prompts with existing LLMs.

The aforementioned divide-and-conquer framework is straightforward for long text processing, and has been explored in previous studies (Chase, 2022; Zhao et al., 2024; Zhang et al., 2024b). However, in our experiments, we find that simply combining an LLM and the divide-and-conquer strategy can not achieve satisfying performance on modern long-text benchmarks (Zhang et al., 2024a).

The major challenge is that segmenting the entire document may disrupt crucial long-range clues. The disrupted long-range information can be divided into two categories: inter-chunk dependencies and inter-chunk conflicts. We therefore focus on enhancing the capabilities of divide-and-conquer frameworks to process cross-chunk information. Specifically, we propose a structured information protocol to address inter-chunk dependencies and in-context confidence calibration to resolve inter-chunk conflicts.

3.3 Structured Information Protocol

An important research question for divide-and-conquer long-text processing frameworks is to determine *what information the map stage should convey to the reduce stage*. If the mapped results are overly simplified, as seen in LongAgent (Zhao et al., 2024), crucial details needed for subsequent stages may be lost. On the other hand, if the mapped results are too complex, they introduce significant computational overhead, increasing the overall latency of the framework.

To this end, we introduce a specialized information structure consisting of four components:

- **Extracted Information**: key facts or data relevant to the query Q that are extracted from the current chunk, providing the necessary

background for subsequent stages to address inter-chunk dependencies.

- **Rationale**: the analysis or inference process that explains how the model derives the intermediate answer from the extracted information, helping to mitigate the risk of hallucinations in subsequent stages.

- **Answer**: the intermediate answer to the query, derived from the extracted information and rationale. If, after providing the rationale, the model determines that the passage does not contain relevant information to address the question, it will output "`NO INFORMATION`", which will be disregarded in subsequent stages.

- **Confidence Score**: a score (out of 5) reflecting the model's confidence in the answer, indicating the completeness and reliability of the information. The confidence score is important for resolving inter-chunk conflicts.

To maintain a consistent input format for the reduce stage, both the map and collapse stages produce data in the structured format described above. A remaining issue with the structured information protocol is the potential **inconsistency in confidence scores** estimated across different chunks when resolving inter-chunk conflicts. Without a general criterion for confidence estimation, the model may assign varying confidence levels to different chunks, even when the content is equally reliable. We thus propose an in-context confidence calibration mechanism to align the confidence scores of different chunks to a consistent standard.

3.4 In-Context Confidence Calibration

To make confidence scores comparable across chunks, we propose calibrating them through in-context learning without adjusting the model parameters. Specifically, we provide confidence estimation principles alongside a typical example for different levels of confidence score. By referencing these principles and examples, the model learns to apply consistent criteria when processing chunks. We can customize different calibration principles and instances for various tasks. Claims fully supported by the provided text are assigned high confidence, while those inferred by the model receive medium confidence. Claims not related to the provided text are assigned low confidence. Figure 2 in provides an example of the calibration prompt. We also provide a prompt example for the map, collapse, and reduce stages in Appendix E. Furthermore, to clearly demonstrate how confidence scores address inter-chunk conflicts, we provide an illustrative example in Appendix F.

```
Assign a confidence score (out of 5) to
your answer based on the completeness and
reliability of the extracted information
and your rationale. The following is some
assigning scoring cases:
<Text: [ Jerry is 18 years old this year. He can
swim and wants to be an athlete. ].
Examples of confidence estimation: [
Jerry can swim, 5 points;
Jerry will become an athlete in the future, 3.5
points;
Jerry will become a swimming athlete in the fu-
ture, 3 points;
Jerry is strong, 3 points;
Jerry can play chess, 0 points;
Jerry likes talking, 0 points ] >.
```

Figure 2: Prompt for in-context confidence calibration.

4 Experiments

4.1 Setup

Models We use two well-known open-source models to validate the effectiveness of the proposed LLM×MapReduce framework, which are Llama3-70B-Instruct (Grattafiori et al., 2024) and Qwen2-72B-Instruct (Yang et al., 2024). We employ vLLM (Kwon et al., 2023) for model inference, and the decoding temperature is set to 0.7.

Evaluation We evaluate the performance of the involved models and methods on InfiniteBench (Zhang et al., 2024a), where the average input length exceeds 100K tokens. This benchmark assesses the long-text capabilities of LLMs across several dimensions, including long-range retrieval, language comprehension, code understanding, and mathematical problem-solving. We exclude the subsets `Code.Run` and `Math.Calc`, as nearly all models achieve less than 5% accuracy on these tasks, making it difficult to differentiate performance among the models. We utilize the evaluation code open-sourced from Zhang et al. (2024a) by default. We find that the recall score for this task tends to increase with longer model outputs. Therefore, we directly engage two human experts with experience in natural language processing to manually assess the accuracy.

Methods	Re.Pa	Re.Nu	Re.KV	En.Sum	En.QA	En.MC	En.Dia	Co.De	Ma.Fi	Avg.
Closed-Source Models										
GPT-4⋆	**100.00**	**100.00**	89.00	14.73	22.44	68.12	7.50	54.31	60.00	57.34
Claude 2⋆	97.80	99.15	65.40	14.50	11.97	67.25	43.00	33.24	32.29	51.62
Kimi-Chat	99.32	97.45	69.20	29.94	18.81	79.91	15.50	38.32	18.57	51.89
Open-Source Models										
YaRN-Mistral⋆	92.71	58.31	0.00	9.09	9.55	29.26	4.50	23.60	17.14	27.13
Yi-6B-200K⋆	**100.00**	94.92	0.00	0.92	9.20	36.68	1.50	18.78	4.29	29.59
Yi-34B-200K⋆	**100.00**	**100.00**	0.00	1.33	12.17	46.29	3.50	21.32	25.71	34.48
Q2-72B-I	**100.00**	**100.00**	29.00	31.85	21.97	81.66	23.00	45.43	59.71	54.74
Divide-and-Conquer Frameworks										
L3-70B-I+LA	99.32	93.05	74.60	2.19	**35.41**	69.00	7.50	24.11	79.14	53.81
L3-70B-I+CoA	9.32	15.59	1.80	10.10	7.03	27.51	9.50	18.27	44.57	15.97
L3-70B-I×MR	**100.00**	99.79	**98.89**	30.63	34.70	82.10	17.50	**62.94**	**91.43**	**68.66**
Q2-72B-I×MR	**100.00**	**100.00**	98.80	**32.39**	23.13	**83.84**	**46.50**	54.82	54.29	65.97

Table 1: Results on InfiniteBench. "⋆" indicates that we directly use the model outputs released by Zhang et al. (2024a) and re-calculate the score. "Q2-72B-I" and "L3-70B-I" refer to Qwen2-72B-Instruct and Llama3-70B-Instruct, respectively. "LA" and "CoA" denote LongAgent (Zhao et al., 2024) and Chain-of-Agents (Zhang et al., 2024b), which are two recent representative frameworks for divide-and-conquer long-sequence processing .

Baselines We select several representative models and methods as our baselines. For closed-source models, we compare against GPT-4, Claude 2 (Anthropic, 2023), and Kimi-Chat. For open-source models, we include YaRN-Mistral (Peng et al., 2024), Yi-6B-200K, Yi-34B-200K (Young et al., 2025), and Qwen2-72B-Instruct. Additionally, we compare LLM×MapReduce with two recent representative frameworks for divide-and-conquer long-sequence processing: LongAgent (Zhao et al., 2024) and Chain-of-Agents (Zhang et al., 2024b).

4.2 Main Results

Table 1 presents the performance of the methods involved on InfiniteBench. For divide-and-conquer methods, the backbone model used is Llama3-70B-Instruct, which has an effective context length of 8K, significantly shorter than the test examples in InfiniteBench. The results indicate that LongAgent (Zhao et al., 2024) outperforms CoA on nearly all subtasks. The proposed LLM×MapReduce method achieves the highest average score, outperforming both the closed-source models and the divide-and-conquer baselines. Augmented by our method, Llama3-70B-Instruct performs well on all the subtasks. Our method is also compatible with Qwen2-72B-Instruct, demonstrating its generalization capability.

4.3 Ablation Study

In LLM×MapReduce, we introduce a structured information protocol and an in-context confidence calibration mechanism, setting our method apart from existing divide-and-conquer baselines. We conduct ablation experiments to investigate the effect of the two components. As shown in Table 2, removing the in-context confidence calibration mechanism leads to a performance decline across all tasks, particularly in English language understanding tasks (i.e., En.Avg). When both confidence calibration and the structured information protocol are disabled, the performance drops even more significantly compared to the full framework. These results underscore the importance of both mechanisms in maintaining strong performance for long-sequence processing.

Method	Re.Avg	En.Avg	Co.De	Ma.Fi
L3-70B-I×MR	99.56	41.23	62.94	91.43
-Conf.	96.00	39.18	58.12	90.00
-Struc.	97.14	25.93	46.45	56.00

Table 2: Effect of structured information protocol and in-context confidence calibration. "Re.Avg" and "En.Avg" denote the average performance on retrieval tasks and English language understanding tasks, respectively.

4.4 Extremely Long Evaluation

Needle-in-a-haystack (NIAH) (Kamradt, 2023) is a widely-used method for evaluating the ability of LLMs to handle long texts by identifying specific facts within long documents. To assess the performance of our framework in handling extremely

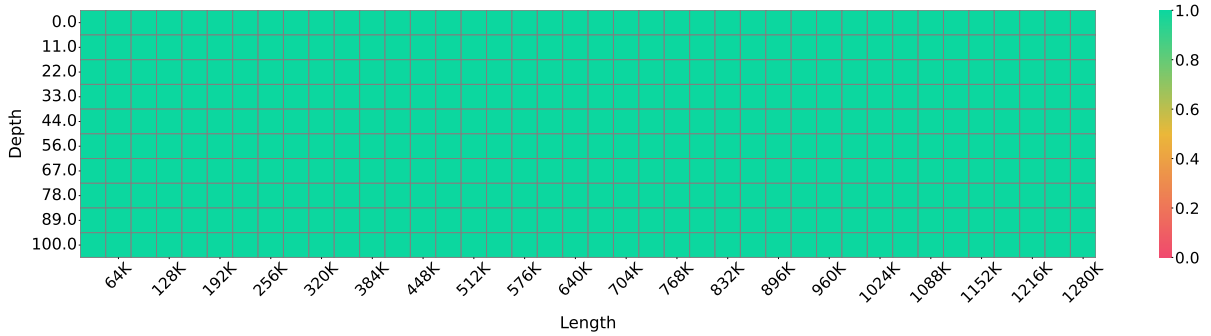

Figure 3: Performance of Llama3-70B-Instruct×MapReduce on the 1280K NIAH test.

long texts, we extend the NIAH test to a length of 1280K tokens. Figure 3 presents the results, demonstrating that our proposed method enables Llama3-70B-Instruct, with a trained context length of 8K tokens, to effectively handle sequences of up to 1280K tokens. This demonstrates the potential of our framework for processing extremely long sequences.

4.5 Inference Latency

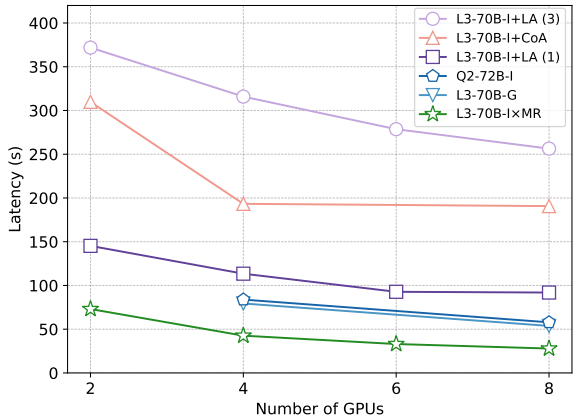

Figure 4: Comparison of inference latency. "L3-70B-G" represents Llama3-70B-Instruct-Gradient-1048K.

Since divide-and-conquer long-sequence processing frameworks introduce multiple intermediate steps, they may be slower than standard decoding. We thus measure the inference latency of the different approaches using 20 test examples, each with 128K tokens. Since the original Llama3-70B-Instruct does not support 128K tokens, we use Llama3-70B-Instruct-Gradient-1048K (Pekelis et al., 2024), an extended version of Llama3-70B-Instruct, to evaluate the inference speed. We report the latency for LongAgent with the maximum number of turns set to 1 and 3. The experiments are conducted using NVIDIA A100 GPUs (80 GB). As shown in Figure 4, both CoA and LongAgent are slower than standard decoding across different settings. However, a notable advantage of divide-and-conquer methods is their lower GPU requirements for handling long sequences. For standard decoding, at least 4 GPUs are needed to process 128K tokens, whereas divide-and-conquer methods can support 128K tokens using just 2 GPUs. Surprisingly, the proposed LLM×MapReduce framework outperforms not only other divide-and-conquer baselines in speed but also standard decoding. The efficiency of our method is achieved by avoiding the need to repeatedly process text chunks to resolve conflicts, as required in LongAgent. Instead, we employ a structured information protocol and an in-context confidence calibration mechanism to effectively integrate information across chunks.

5 Pyramid-Align: Let Short Teach Long

An additional advantage of divide-and-conquer methods is that they allow short-context LLMs to generate long-context supervised fine-tuning (SFT) data. In other words, we can distill the capabilities of short-context LLMs into long-context LLMs, achieving the goal of "letting short teach long". In this work, we adopt this idea to create a long SFT dataset called Pyramid-Align. Several existing studies utilize LLMs to generate questions based on long documents. LongAlign (Bai et al., 2024) proposes using Claude 2.1 to ask questions conditioned on the entire text. In contrast, IN2 (An et al., 2024) provides a local chunk to the LLM for question generation before reintegrating that chunk back into the long document. Instead of relying solely on the entire document or a specific chunk, we propose generating questions based on varying amounts of information. Intuitively, a high-quality long SFT dataset should train the model to thoroughly comprehend any span within the document.

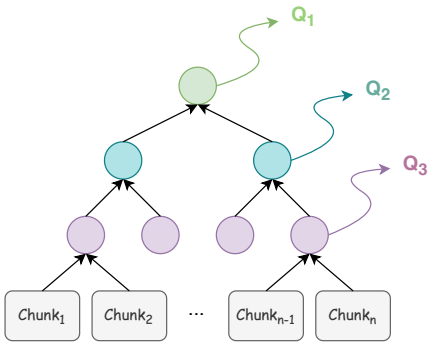

Figure 5: Illustration of Pyramid-Align.

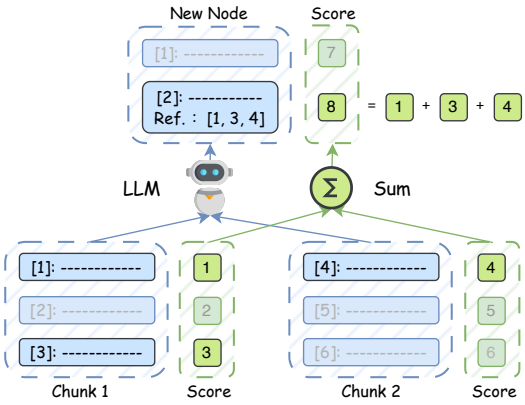

Figure 6: Propagation of the sentence-level importance score from leaf nodes to upper nodes.

Constructing the Pyramid As illustrated in Figure 5, we utilize LLMs to abstract chunks into hierarchical nodes, where nodes at different levels encompass varying amounts of information. Specifically, each leaf node represents an individual chunk, while higher-level nodes contain progressively more information spanning multiple chunks. To form these higher-level nodes, we typically combine three lower-level nodes (adjusting downwards if necessary to fit in the context window L) and prompt the LLM to synthesize their content. The number of levels is dynamically determined by document length. Longer documents result in deeper hierarchies. In our dataset, the height ranges from 1 to 3, with an average of 1.99.

Improving Information Coverage The primary challenge in constructing the pyramid is that nodes at higher levels may lose essential content, resulting in reduced informativeness as the hierarchy ascends. Improving the information coverage while summarizing effectively at each level is crucial to ensure that upper-level nodes retain sufficient context and details to support accurate understanding across the entire document. To this end, we design a bottom-up importance propagation mechanism, where the sentence-level importance scores help the LLM identify key content. Figure 6 illustrates the propagating procedure. When constructing upper nodes, we prompt the LLM to pay more attention to sentences with higher importance scores. Please refer to Section B in Appendix for more details.

Generating Questions After constructing the pyramid, we randomly select nodes to generate questions. Selecting leaf nodes mimics IN2 (An et al., 2024) while selecting the root node aligns with LongAlign (Bai et al., 2024). Additionally, the LLM is guided to prioritize important sentences when formulating questions, ensuring that key information is considered.

Generating Answers Given a long document X and a question Q, we employ the LLM×MapReduce framework to generate the answer A. Each example in the Pyramid-Align dataset is represented as (X, Q, M, A), where M denotes the intermediate outputs (e.g., the mapped and collapsed results) from the LLM×MapReduce process. This format allows us to not only use (X, Q, A) to learn a standard long-context LLM, but also leverage the intermediate outputs, namely (X, Q, M, A), to train a model that aligns more closely with the LLM×MapReduce framework.

Dataset Construction To ensure a fair comparison between different data construction methods, we use the same documents used in previous works. Specifically, we extract all the 5,299 English documents from the LongAlign dataset. The average document length is approximately 17K tokens, with a maximum length of 74K tokens. For each document, we generate question-answer pairs using the method described above. This process resulted in a final dataset of 4,927 entries.

Standard Long-Context Training To validate the effectiveness of our data synthesis approach, we use the same backbone model, Llama3.1-8B, to train long-context models on both LongAlign and Pyramid-Align. Note that both long SFT datasets are derived from the same set of long documents, and the trained models perform standard decoding without LLM×MapReduce. Since we only use English documents, we evaluate the trained models on four English tasks from InfiniteBench. As shown in Table 3, the model trained with Pyramid-

Model	Sum	QA	MC	Dia	Avg.
Baseline					
GPT-4	14.73	22.44	68.12	7.50	28.20
Standard Decoding					
L (LA)	16.46	6.17	44.54	10.50	19.42
L (PA)	25.22	19.91	43.67	11.00	**24.95**
LLM×MapReduce					
L (PA)	28.55	21.22	67.69	10.00	**31.86**

Table 3: Comparison between LongAlign and Pyramid-Align on the English tasks of InfiniteBench. "L (LA)" denotes the model trained from Llama3.1-8B using LongAlign, while "L (PA)" denotes the model trained from the same backbone on Pyramid-Align.

Align outperforms the one trained with LongAlign, demonstrating the effectiveness of Pyramid-Align.

Training with LLM×MapReduce As aforementioned, we can also leverage (X, Q, M, A) to enhance the ability of LLMs to operate effectively in the MapReduce framework. We train the model from the same backbone (i.e., Llama3.1-8B) using the data in the (X, Q, M, A) format. As shown in Table 3, the trained 8B model can outperform the strong baseline, GPT-4, with the help of our proposed LLM×MapReduce framework.

6 Conclusion

We introduce LLM×MapReduce, an effective divide-and-conquer framework for long-sequence processing, which can also serve as a powerful data synthesis engine for long-alignment resources. The experimental results validate the effectiveness of our approach, surpassing standard long-context LLMs and other divide-and-conquer baselines.

Limitations

In this paper, we present the LLM×MapReduce framework, offering a general solution for processing long texts, particularly for standard document types. However, the current implementation may not fully accommodate the unique requirements of specialized formats, such as visually rich academic papers containing diagrams or other multi-modal elements. Furthermore, the document chunking mechanism may face challenges when processing unstructured texts that lack clear paragraph boundaries. Future work could focus on developing adaptive chunking algorithms and expanding the framework to better support domain-specific tasks.

Acknowledgments

We thank all anonymous reviewers for their valuable comments and suggestions on this work. This work is supported by National Science and Technology Major Project (Grant No. 2022ZD0116101), the Major Scientific Research Project of the State Language Commission in the 13th Five-Year Plan (Grant No. WT135-38), and the public technology service platform project of Xiamen City (No. 3502Z20231043). This work is also supported by the AI9Stars community.

References

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, and Jian-Guang Lou. 2024. Make your llm fully utilize the context. *Preprint*, arXiv:2404.16811.

Anthropic. 2023. Model card and evaluations for claude models.

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024. Longalign: A recipe for long context alignment of large language models. *Preprint*, arXiv:2401.18058.

Harrison Chase. 2022. Langchain. `https://github.com/langchain-ai/langchain`.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. *Preprint*, arXiv:2309.12307.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Greg Kamradt. 2023. Llms need needle in a haystack test-pressure testing llms.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *Preprint*, arXiv:2308.09583.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2024. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Leonid Pekelis, Michael Feil, Forrest Moret, Mark Huang, and Tiffany Peng. 2024. Llama 3 gradient: A series of long context models.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Yujia Zhou, Xu Chen, and Zhicheng Dou. 2024. Are long-llms a necessity for long-context tasks? *Preprint*, arXiv:2405.15318.

Wei Shi, Shuang Li, Kerun Yu, Jinglei Chen, Zujie Liang, Xinhui Wu, Yuxi Qian, Feng Wei, Bo Zheng, Jiaqing Liang, Jiangjie Chen, and Yanghua Xiao. 2024. SEGMENT+: Long text processing with short-context language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16605–16617, Miami, Florida, USA. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Shengnan Wang, Youhui Bai, Lin Zhang, Pingyi Zhou, Shixiong Zhao, Gong Zhang, Sen Wang, Renhai Chen, Hua Xu, and Hongwei Sun. 2024. Xl3m: A training-free framework for llm length extension based on segment-wise inference. *Preprint*, arXiv:2405.17755.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024a. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *Preprint*, arXiv:2402.04617.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024b. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. 2023. Effective long-context scaling of foundation models. *Preprint*, arXiv:2309.16039.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, et al. 2024. Qwen2 technical report.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li,

Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yanpeng Li, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. 2025. Yi: Open foundation models by 01.ai. *Preprint*, arXiv:2403.04652.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024a. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, Bangkok, Thailand. Association for Computational Linguistics.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arik. 2024b. Chain of agents: Large language models collaborating on long-context tasks. *Preprint*, arXiv:2406.02818.

Jun Zhao, Can Zu, Hao Xu, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024. Longagent: Scaling language models to 128k context through multi-agent collaboration. *Preprint*, arXiv:2402.11550.

A Effect of Chunk Size

To examine the impact of chunk size, we assess our framework on the En.QA task from InfiniteBench using chunk sizes ranging from 0.5k to 6k tokens. The model used in this evaluation is Llama3-70B-Instruct, with all other experimental settings consistent with those described in Section 4. As shown in Figure 7, increasing the chunk size generally leads to improved performance, suggesting that larger chunks provide more comprehensive contextual information, which benefits the model's ability to comprehend and answer questions accurately. However, the performance gain diminishes as chunk sizes increase, suggesting a trade-off between contextual completeness and the increased difficulty of processing intensive information.

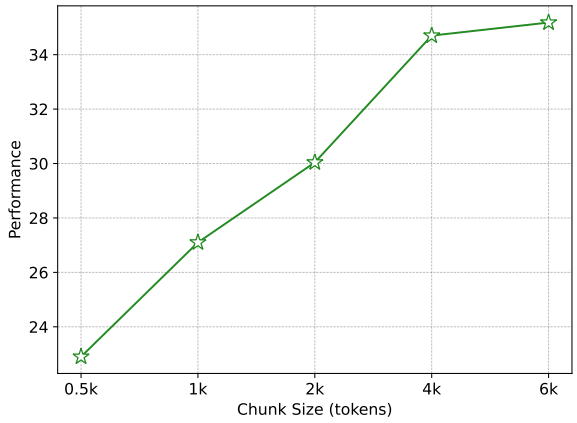

Figure 7: Effect of the chunk size. Results are reported on the En.QA task from InfiniteBench.

B Details of Importance Propagation

During the construction of Pyramid-Align, we assign an importance score and an index for each sentence within the pyramid. The importance scores help the LLM identify key content, while the indices are used to propagate the scores in a bottom-up manner. For the leaf nodes, the sentence-level importance score is calculated as the sum of the TF-IDF scores of the words in the sentence. For higher-level nodes, we require the LLM to record the sentences from the lower-level nodes using the sentence indices. The score of each sentence at a higher level is then the sum of the importance scores of the corresponding lower-level sentences. Figure 6 shows an example.

C Details of the Construction of Pyramid-Align

We construct the Pyramid-Align dataset using Qwen2-72B-Instruct-AWQ (Yang et al., 2024), leveraging vLLM with a decoding temperature of 0.7. The dataset is built from all the 5,299 English documents in the LongAlign dataset, which are structured into a pyramid format. After constructing the pyramid for each document, a single node is randomly selected as the context, and one question is generated from it, chosen from three categories: causal reasoning, information extraction, or summarization. Figure 8 illustrates the prompt used for generating causal reasoning questions.

We then generate answers using the proposed LLM×MapReduce framework, powered by Llama3-70B-Instruct, resulting in 5,299 question-answer pairs. After filtering out pairs with illegal characters, the final dataset contains 4,927 instances. On average, the dataset has a document length of 17K tokens, a question length of 23 tokens, and a response length of 151 tokens, as measured with the LLaMA 3 tokenizer.

```
Propose a question based on the given
text. Bold words require extra attention
when asking questions. This question
must be about reasoning, such as sort,
timeline arrangement, and cause-effect
relationship identification. Sorting
involves organizing data in a specific
order, timeline arrangement refers to
placing events in chronological order, and
cause-effect relationship identification
is the process of determining how one event
or action can directly lead to another.

### Text:

{context}

### Question:
```

Figure 8: Prompt for generating causal reasoning questions during the construction of Pyramid-Align.

D Details of Model Training

Following LongAlign, we mix our dataset with ShareGPT (Chiang et al., 2023) data for training. Specifically, for standard training, we combine the 4,927 generated Pyramid-Align instances with the ShareGPT data. Similarly, we extract the corresponding 4,927 instances from LongAlign

and mix them with ShareGPT to create a comparable dataset. Both datasets are used to train long-context models on the same backbone model, Llama3.1-8B, for 1,500 steps (approximately 2 epochs), with a learning rate of 2e-5. For training with LLM×MapReduce, we leverage data in the (X, Q, M, A) format to enhance the ability of LLMs to perform effectively within the proposed framework. We combine this data with ShareGPT to create a mixed dataset and train the same model (i.e., Llama3.1-8B) for 2 epochs with a learning rate of 2e-5. All experiments are conducted on 32 NVIDIA A100 80G GPUs.

E Prompt Example

To better understand the LLM×MapReduce framework, we provide the prompts for general question answering as an example. Specifically, Figure 9 shows the prompt for the map stage, Figure 10 presents the prompt for the collapse stage, and Figure 11 displays the prompt for the reduce stage. Note that "{map result}" refers to the concatenated structured information generated during the map stage within each group.

F Inter-Chunk Conflict and Dependency Resolution Example

To better illustrate how our method resolves inter-chunk conflicts, we provide the following explanation and example. Figure 12 shows an example question for the LLM to answer. Table 4 lists the key information extracted from each relevant chunk, the potential answer derived solely from that chunk, and its associated confidence score. Both Chunk 0 and Chunk 1 strongly indicate that Cartwright was searching through the garbage (B), with confidence scores of 3.5 and 4, respectively. In contrast, Chunk 2 only mentions newspaper stands without confirming that Cartwright searched them, leading to a lower confidence of 1.5. Given this, the model correctly resolves the conflict and selects **B. The garbage** as the final answer. This example demonstrates how our approach integrates structured information, assesses inter-chunk dependencies, and resolves conflicting inferences using confidence scores, ensuring a more reliable final decision.

Chunk	**Extracted Info.**	**Ans. & Conf.**
Chunk 0	Cartwright visits 23 hotels, and Giorgio suspects the cut-up newspaper might be in their waste-paper baskets.	Garbage (B) Confidence: 3.5
Chunk 1	Giorgio instructs Cartwright to check "yesterday's waste-paper," reinforcing the search in trash.	Garbage (B) Confidence: 4
Chunk 2	Cartwright visits 23 hotels but does not find the newspaper. Some hotels have small newspaper stands outside.	Newspaper stand (A) Confidence: 1.5 (weak inference)

Table 4: Example data showing chunk information and confidence scores used for conflict resolution. "Extracted Info." and "Ans. & Conf." denote Extracted Information and Answer & Confidence, respectively. The example shown here has been condensed for brevity due to space constraints.

```
You are provided with a portion of an article and a question. Read the article portion and follow my instructions to process it.
Article:
The article begins as follows:
{context}
The article concludes here.
Question:
{question}
Instructions:
Please extract information from the provided passage to try and answer the given question. Note that you only have a part of the entire text, so the information you obtain might not fully answer the question. Therefore, provide your rationale for using the extracted information to answer the question and include a confidence score. The following is some assigning scoring cases: <Text: [Jerry is 18 years old this year. He can swim and wants to be an athlete.]. assigning scoring: [Jerry can swim, 5 points; Jerry will become an athlete in the future, 3.5 points; Jerry will become a swimming athlete in the future, 3 points;Jerry is strong,3 points; Jerry can play chess, 0 points;Jerry likes talking,0 points]>. Follow these steps:
1. Extract Relevant Information: Identify and highlight the key pieces of information from the passage that are relevant to the given question.
2. Provide a Rationale: Analyze the extracted information and explain how it can be used to address the question. If the information is incomplete, discuss any assumptions or inferences you need to make.
3. Answer the Question: Based on your rationale, provide the best possible answer to the question. If, after providing your rationale, you believe the passage does not contain any information to solve the question, output "[NO INFORMATION]" as the answer.
4. Assign a Confidence Score: Assign a confidence score (out of 5) to your answer based on the completeness and reliability of the extracted information and your rationale process.
Please follow this format:
Extracted Information:
Rationale:
Answer:
Confidence Score:
```

Figure 9: Example for the prompt of the map stage

```
You need to process a task with a long context that greatly exceeds your context limit. The only feasible way to handle this is by processing the long context chunk by chunk. You are provided with a question and some information extracted from each chunk. Each piece of information contains Extracted Information, Rationale, Answer, and a Confidence Score. The following is some assigning scoring cases: <Text: [Jerry is 18 years old this year. He can swim and wants to be an athlete.]. assigning scoring: [Jerry can swim, 5 points; Jerry will become an athlete in the future, 3.5 points; Jerry will become a swimming athlete in the future, 3 points;Jerry is strong,3 points; Jerry can play chess, 0 points;Jerry likes talking,0 points]>. Read the information and follow my instructions to process it.
Extracted Information:
The extracted information begins as follows:
{map result}
The extracted information concludes here.
Question:
{question}
Instructions:
Integrate the extracted information and then reason through the following steps:
1. Integrate Extracted Information: Collect and summarize all the evidence relevant to solving the question. Consider the confidence scores of each piece of extracted information to weigh their reliability. Higher confidence scores should be given more importance in your summary.
2. Analyze: Re-analyze the question based on the summarized information. Use the confidence scores to determine the reliability of different pieces of information, giving more weight to information with higher confidence scores.
3. Answer the Question: Provide the best possible answer based on the updated information. If, after providing your rationale, you believe the passage does not contain any information to solve the question, output "[NO INFORMATION]" as the answer. Use the confidence scores to support the reliability of your final answer, prioritizing higher confidence information.
4. Assign Confidence Score: Give a confidence score (out of 5) for your final answer based on the completeness and reliability of the updated information and your rationale process.
Consider the initial confidence scores of the integrated information to determine your final confidence score.
Please follow this format:
Extracted Information:
Rationale:
Answer:
Confidence Score:
```

Figure 10: Example for the prompt of the collapse stage.

```
You need to process a task with a long context that greatly exceeds your context limit. The only feasible way to handle this is by processing the long context chunk by chunk. You are provided with a question and some information extracted from each chunk. Each piece of information contains Extracted Information, Rationale, Answer, and a Confidence Score. The following is some assigning scoring cases: <Text: [Jerry is 18 years old this year. He can swim and wants to be an athlete.]. assigning scoring: [Jerry can swim, 5 points; Jerry will become an athlete in the future, 3.5 points; Jerry will become a swimming athlete in the future, 3 points;Jerry is strong,3 points; Jerry can play chess, 0 points;Jerry likes talking,0 points]>. Read the information and follow my instructions to process it.
Question:
{question}
Information from chunks:
{collapse result}
Each chunk provides extracted information related to the same question, but due to partial data, conclusions from each chunk might vary. Your role is to integrate and reason through this information, weighing confidence scores to resolve any inconsistencies. Then provide the final answer.
Please follow this format:
Rationale:
Answer:
```

Figure 11: Example for the prompt of the reduce stage

Question: Where does Giorgio send Cartwright in search of the cut-up newspaper?

Options:

A. A newspaper stand
B. The garbage
C. The printer's
D. Devonshire

Figure 12: The question and options for our example illustrating inter-chunk dependency and conflict resolution. Refer to Table 4 for the associated chunk information and confidence scores.