Lost-in-the-Middle in Long-Text Generation: Synthetic Dataset, Evaluation Framework, and Mitigation Paradigm

Anonymous ACL submission

Abstract

Existing long-text generation methods primarily focus on creating lengthy outputs from short inputs, but neglect the critical challenge of long-input-to-long-output generation. In scenarios with long input sequences, important information in the middle of such sequence may be overlooked, a problem commonly known as the "lost-in-the-middle" phenomenon. This phenomenon becomes more pronounced as the input length increases, leading to inconsistencies and incoherence in the generated output. To address this, we propose **R**etrieval-**A**ugmented **L**ong-Text **Writer** (RAL-WRITER), which consists of a *Planner* that generates writing steps and a *Writer* that produces content based on these steps. The *Writer* incorporates a mechanism to compute an importance score to dynamically retrieve and strategically restate critical input segments by jointly modeling semantic relevance and positional bias. We also construct the first dataset for long-input generation and introduce three evaluation metrics covering length, consistency, and quality. We use this dataset to evaluate our RAL-WRITER against comparable baselines. The results demonstrate the effectiveness of our approach [1].

1 Introduction

Although long-context LLMs demonstrate a strong ability to process extensive inputs, they often fail to generate outputs matching the length specified in instructions. As reported by Bai et al. (2025), when asked to produce a 10,000-word paper, LLMs typically generate fewer than 2,000 words. This limitation arises because LLMs are rarely exposed to long-form outputs during supervised fine-tuning. Prior work addresses this issue through multi-step agent frameworks (Xiong et al., 2025), long-response SFT (Bai et al., 2025), and preference alignment methods (Pham et al., 2024).

[1]Code is publicly available at https://anonymous.4open.science/r/RAL-Writer-5358

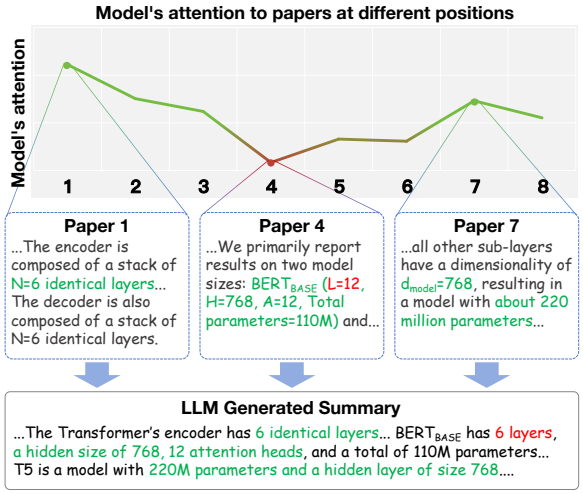


Figure 1: The curve illustrates how an LLM's attention distribution changes across a long context. It is obtained from our empirical experiments (detailed in §6.7). In long-text generation tasks, the LLM is more prone to errors when handling information located in the middle part of the input.

However, real-world applications require long-form generation conditioned on long inputs, such as report generation from extensive logs (Xu et al., 2024; Astekin et al., 2024), document-level summarization (Fang et al., 2024; Van Schaik and Pugh, 2024), and article continuation (Xie et al., 2023; Gurung and Lapata, 2025). Despite its practical importance, the task of generating long outputs from long inputs (long-input-to-long-output) remains underexplored and presents significant challenges in both long-context understanding and sustained text generation.

Compared to long-form generation tasks guided by short instructions, the setting of long-input-to-long-output generation reveals a more pronounced challenge. Prior studies show that with lengthy inputs, LLMs often suffer the "lost-in-the-middle" phenomenon (Liu et al., 2024a), overlooking essential information in intermediate positions. In

Method	Long Input (> 8K)	Long Output (> 1K)	Real-world Aligned	Consistency Evaluation	Quality Evaluation
NIAH (Kamradt, 2023)	✓	✗	✗	✓	✗
RULER (Hsieh et al., 2024)	✓	✗	✗	✓	✗
∞Bench (Zhang et al., 2024)	✓	✗	✓	✓	✗
SummHay (Laban et al., 2024)	✓	✗	✓	✓	✗
LongGenBench_1 (Wu et al., 2025)	✗	✓	✗	✗	✗
LongGenBench_2 (Liu et al., 2024b)	✗	✓	✗	✗	✗
LongWriter (Bai et al., 2025)	✗	✓	✓	✗	✓
ProxyQA (Tan et al., 2024)	✗	✓	✓	✓	✗
Ours	✓	✓	✓	✓	✓

Table 1: Recent representative benchmarks for long-context LLM evaluation. "Real-world Aligned" indicates whether tasks reflect practical applications, "Consistency Evaluation" assesses factual correctness, and "Quality Evaluation" measures language fluency and structural coherence.

realistic long-document scenarios, such degradation propagates to generation, hindering faithful output (Figure 1). Consequently, existing benchmarks fail to jointly evaluate input understanding and output generation—most treat them as independent tasks (Table 1), neglecting the core challenge of cohesive long-form reasoning.

To address the "lost-in-the-middle" issue, we propose the **R**etrieval-**A**ugmented **L**ong-Text **Writer** (RAL-WRITER), which explicitly identifies and preserves essential information via an importance-aware retrieval mechanism. RAL-WRITER comprises a *Writing Step Planner* and a *Retrieve-and-Restate Writer*. The *Planner* generates writing steps for long outputs based on extensive input. The *Writer* subsequently sequentially generates content step-by-step. During writing, RAL-WRITER dynamically retrieves mid-sequence chunks with the highest importance score and restates them at the prompt tail to explicitly mitigate attention decay.

For the lack of benchmarks, we introduce a long-form summarization task in which models are required to generate comprehensive summaries of specified length from multiple scientific papers. This task naturally depends on long inputs, involves knowledge-intensive content, and offers practical value. To support systematic evaluation, we manually construct a dataset comprising 100 samples (300 arXiv [2] papers) spanning diverse categories and document lengths.

In addition, we develop an evaluation framework that assesses model performance from three aspects: length, consistency, and quality. The length metric examines whether the generated summaries satisfy specified length requirements. Consistency is evaluated through carefully constructed question–answer (QA) pairs that verify whether critical information from multiple source papers is captured—specifically implemented by an advanced LLM. Finally, quality—including fluency, coherence, and structural organization—is measured using an LLM-as-a-judge approach guided by a comprehensive checklist. Together, these metrics go beyond traditional surface-level measures such as ROUGE or BLEU, enabling a rigorous and multi-faceted assessment of long-text summarization.

Our contributions are summarized as follows:

- We propose RAL-WRITER, a retrieval-based generation framework that restates crucial yet easily overlooked content to mitigate the "lost-in-the-middle" problem. To our knowledge, this is the first work to address this challenge.
- In the absence of benchmarks for long-input and long-output generation, we construct a long-form summarization dataset. We further design a three-dimensional evaluation framework on length, consistency, and quality.
- Extensive experiments on our dataset demonstrate the effectiveness of RAL-WRITER in handling long-context generation tasks.

2 Related Work

2.1 Long-context LLMs

Extended context is crucial for LLMs to leverage longer documents and few-shot examples, but it increases memory and computation. FlashAttention (Dao et al., 2022) reduces memory usage via IO-aware attention, while Block-wise Parallel Transformer (BPT) (Liu and Abbeel, 2023) fuses feedforward and attention layers to handle sequences up to four times longer. Rotary Position Embedding (Su et al., 2024; Peng et al., 2024; Zhu et al., 2024) enables long-context inference even for models not trained on long sequences. Methods such as LM-Infinite (Han et al., 2024), LongLoRA (Chen et al., 2024), and LongQLoRA (Yang, 2023) further extend context size. Despite these advances, it remains unclear whether LLMs fully utilize extended contexts.

[2] https://arxiv.org/

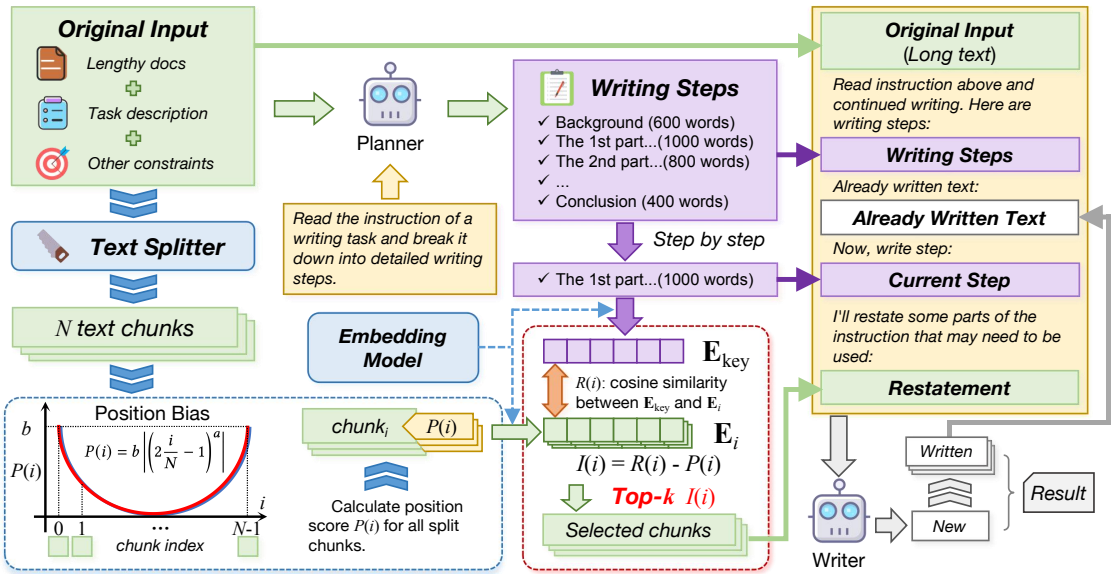


Figure 2: Illustration of RAL-WRITER. The *Planner* decomposes the task into Writing Steps. The Text Splitter divides the input into chunks, and each chunk is individually calculated for the relative position score and the relevance score. The top-k relevant chunks are selected as context for the *Writer*, which generates text step by step following the planned writing structure.

2.2 Long-form Text generation

LLMs have difficulty generating long documents in a single pass. Prior work addresses long-form generation through model-based approaches—Suri-I-ORPO (Pham et al., 2024), LongWriter (Bai et al., 2025), and LongDPO (Ping et al., 2025)—as well as agent-based methods such as Write-HERE (Xiong et al., 2025) and the Generator–Extender framework (Quan et al., 2024).

2.3 Lost-in-the-Middle

Liu et al. (2024a) first identified the "lost-in-the-middle" problem, where LLMs struggle to capture information in the middle of long contexts. Subsequent work addresses this issue via different strategies: An et al. (2024) constructs dense QA data to improve long-document comprehension, He et al. (2024) introduces a Position-Agnostic Multi-step QA task to enhance multi-document QA, and Wu et al. (2024) optimizes position encoding to mitigate attention gaps in the middle.

3 RAL-WRITER

AgentWrite (Bai et al., 2025) demonstrates that long-form generation can be achieved via a "Plan and Write" paradigm, but it primarily targets short inputs. In long-input settings, it is susceptible to the "lost-in-the-middle" problem (§6.7). To mitigate this issue, we propose RAL-WRITER, which composed with a *Planner* for plan generation and a *Retrieve-and-Restate Writer* that generates content. Figure 2 shows the workflow.

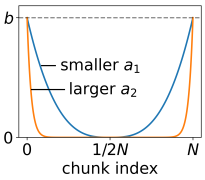
3.1 Writing Steps Planner

An effective strategy for generating long-form content with LLMs is to decompose the task and handle each part separately. The *Planner* plays a crucial role in this decomposition process. It is designed to generate the overall structure and writing steps after comprehending lengthy inputs. Typically, the input consists of a long text and an instruction specifying the desired output length, as shown in Appendix C. After processing the entire input, the *Planner* produces a comprehensive writing plan comprising multiple steps, with each step including specific writing requirements and expected length. The total length of all steps is expected to precisely match the target length specified by the user.

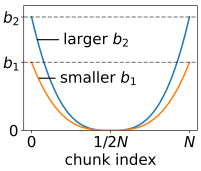
3.2 Retrieve-and-Restate Writer

The *Writer* generates content step by step according to the plan, using a prompt that includes the input, previously written text, and current-step requirements. To prevent loss of important information, we employ a retrieve-and-restate mechanism that retrieves crucial chunks from the long text and incorporates them through strategic restatement.

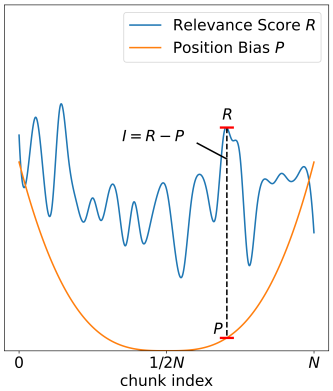
Long-text Chunking. To ensure contextual coherence within each chunk, we implement a recursive splitter following LangChain (Chase, 2022).



(a) P varies with a.



(b) P varies with b.



(c) The schematic of R, P, and I.

Figure 3: (a) A larger a means that the slope of P is greater near both ends, while the middle part is closer to 0. (b) A larger b indicates that P can achieve a greater maximum value at both ends. (c) Importance I is defined as the difference between P and R; the stronger the relevance of a chunk to the current step and the closer its position is to the middle, the greater the value of I becomes.

Especially, it splits long texts into small chunks based on logical structures (such as paragraphs, tables, and lists) and combines adjacent chunks until reaching a preset size. Meanwhile, to avoid information loss after splitting, the merged chunks also have a certain amount of overlap.

Important Chunks Retrieval. As analyzed by Liu et al. (2024a), when the input text is lengthy, the "lost-in-the-middle" issue naturally arises. To explicitly prompt the model to focus on important yet overlooked information, we propose an important chunks retrieval mechanism. Intuitively, a chunk with a higher Relevance Score for a given writing step, yet suffering more severely from the "lost-in-the-middle" issue (i.e., positioned closer to the middle), is considered more crucial.

Formally, denote the i-th chunk embedding as $\mathbf{E}_i$, and the embedding of the current step as $\mathbf{E}_{\text{key}}$, the Relevance Score $R(i)$ can be defined as:

$$R(i) = \frac{\langle \mathbf{E}_i, \mathbf{E}_{\text{key}} \rangle}{||\mathbf{E}_i||\,||\mathbf{E}_{\text{key}}||}. \quad (1)$$

To measure a chunk's relative position within the context, we introduce a Relative Position Bias for each chunk. Chunks near the ends of the input are more easily attended to, while those in the middle are often overlooked. Relative Position Bias is designed to satisfy three properties: it peaks at both ends, reaches a minimum at the center, and is symmetric about the midpoint. Formally, we define:

$$f(x) = b|(2x-1)^a|, \quad (2)$$

where $x \in [0,1]$ is the normalized position and $a, b > 0$ are tunable. The transformation $(2x-1)$ ensures symmetry, the absolute value sets minima at the center and maxima at the edges, and a controls the steepness while b sets the maximum score. For example, when $a = 1$, the function decreases linearly toward the center, while larger values of a yield a sharper drop, emphasizing the severity of the "lost-in-the-middle" effect.

To apply this formulation to discrete chunks, we consider an input divided into N chunks. The relative position of the i-th chunk can be written as $x = i/N$. Substituting this into the function, the Position Bias becomes:

$$P(i) = f\left(\frac{i}{N}\right) = b\left|\left(2\frac{i}{N} - 1\right)^a\right|. \quad (3)$$

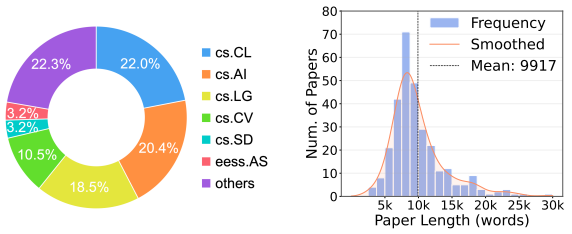
This definition naturally links the chunk position to the observed "lost-in-the-middle" trend. By adjusting the parameters a and b, we can flexibly model different degrees of positional bias, enabling the framework to better reflect the empirical tendencies of language models to overlook central chunks.

Consequently, the Importance Score I for each chunk is defined as the difference between the Relevance Score R and the Position Bias P:

$$I(i) = R(i) - P(i). \quad (4)$$

Chunks with higher Importance Scores are reinforced in the prompt to guide the *Writer*'s attention. We retrieve the top-k chunks ranked by Importance Score I for the subsequent restatement stage. Figure 3 visualizes the resulting Relevance and Position Bias curves for an example.

Restatement of Retrieved Chunks. The retrieved chunks will be embedded in the prompt of the *Writer* LLM during the writing phase. These text chunks are systematically concatenated at the tail of the input prompt, a strategic placement designed to amplify the LLM's attention allocation toward the appended content. This architectural choice capitalizes on the positional sensitivity inherent in transformer-based attention mechanisms, where later input segments typically receive heightened computational prioritization during token prediction. Noted, we sort the retrieved chunks in ascending order of their Importance Scores, ensuring that the more important chunks are positioned closer to the end, minimizing the risk of being lost.



(a) Category distribution (b) Length distribution

Figure 4: Statistical information of our dataset. Noted, categories in (a) with proportions less than 3% are grouped into "Others".

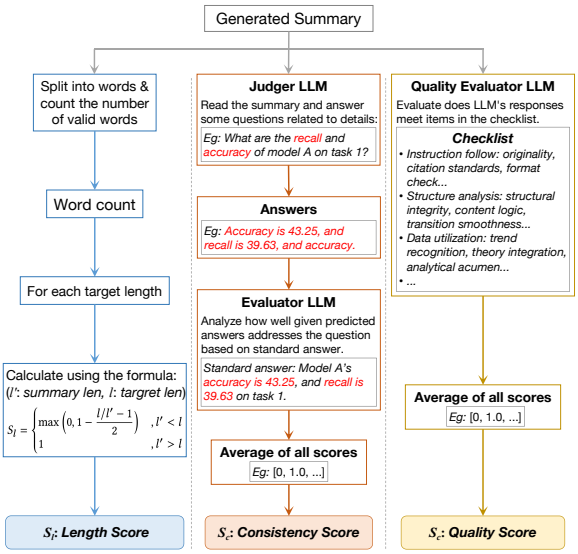


Figure 5: Overview of our evaluation framework.

4 Dataset Synthetic

To effectively assess the model's ability for long-text understanding and generation, we construct a dataset in which the task involves generating a long scientific paper summary (according to the survey's structural requirements, task instruction is shown in Appendix B) by reading extensive inputs consisting of multiple papers on related topics. We chose this task for the following reasons:

- Combining multiple papers can easily produce input lengths of tens of thousands of words, providing a rigorous test of the model's capacity to process extensive content.
- Scientific papers are a representative form of knowledge-intensive text, making them well-suited for evaluating the model's ability to manage and integrate complex information.
- Long-text summaries of scientific papers can substantially alleviate the cognitive burden of understanding the original documents, offering practical value to researchers.

Each sample comprises a long input formed by concatenating multiple scientific papers on a shared topic, often spanning tens of thousands of words. During evaluation, models generate long-form summaries from these inputs, assessing their ability to integrate and distill knowledge from lengthy texts into coherent narratives.

For each sample in the dataset, we collect a triplet of thematically related papers from arXiv, typically those that are introduced within the same survey or compared in the same research work. We download the corresponding LaTeX source files and preprocess the data by removing extraneous elements such as comments, preambles, and appendices. To facilitate structural comprehension by large language models, we retain essential LaTeX markup in the cleaned text. We further exclude papers with inconsistent formatting or insufficient length to ensure data quality. The resulting dataset consists of 100 samples (300 papers in total). Figure 4 summarizes key characteristics of the curated dataset, including the distribution of arXiv categories and document lengths.

5 Evaluation

To assess the generated summaries, we introduce three complementary metrics: length score, consistency score, and quality score. These metrics are designed to capture different aspects of long-text summarization performance, ensuring a comprehensive evaluation. Figure 5 provides an overview of the task workflow and the evaluation framework.

5.1 Length Evaluation

Whether the length of generated summaries meets the requirements is a key metric for long-text generation. Inspired by LongBench-Write (Bai et al., 2025), we utilize a linear piecewise function to evaluate the length score S_l, defined as follows:

$$S_l = \begin{cases} \max\left(0, 1 - \dfrac{l/l' - 1}{2}\right) & , l' < l \\ 1 & , l' \geq l \end{cases} \quad (5)$$

where l' is the length of generated summary, and l is the required length. When $l \geq l'$, the score attains its max value of 1, indicating that the generated summary has sufficient length as required. When l' is lower than l, the score diminishes.

Backbone	Method	Overall			4k			8k			16k		
		S_l	S_c	S_q	S_l	S_c	S_q	S_l	S_c	S_q	S_l	S_c	S_q
GPT-4o	Single	0.00	23.14	56.06	0.00	24.77	57.52	0.00	22.91	55.87	0.00	21.73	54.79
LongWriter-glm4-9b	Single	40.18	35.29	72.79	79.85	33.17	71.90	39.15	36.71	72.48	1.53	36.00	74.00
Qwen2.5-14B	Single	6.92	40.12	66.73	19.41	38.02	65.03	1.34	40.83	67.32	0.00	41.52	67.84
	AgentWrite	**79.25**	54.10	74.49	**100.00**	51.35	74.22	**93.90**	55.66	74.51	**43.86**	**55.15**	74.75
	Compress	64.33	44.49	74.01	**100.00**	43.67	74.87	84.84	44.85	74.24	8.14	44.94	72.93
	RAL-WRITER	75.15	**55.28**	**75.68**	**100.00**	**53.17**	**75.49**	87.07	**58.23**	**75.93**	38.39	54.43	**75.62**
Qwen2.5-32B	Single	4.90	40.46	66.63	14.63	39.06	66.28	0.06	40.69	66.00	0.00	41.63	67.62
	AgentWrite	61.34	52.39	74.11	97.18	49.98	73.19	74.18	54.08	74.79	12.67	53.10	74.34
	Compress	52.28	46.47	73.69	92.46	43.31	73.33	60.50	50.17	74.03	3.88	45.94	73.70
	RAL-WRITER	**77.09**	**54.15**	**75.67**	**99.83**	**53.77**	**74.68**	**91.22**	**55.54**	**76.83**	**40.22**	**53.15**	**75.51**

Table 2: The main results on our dataset. S_l, S_c, and S_q stand for the length score, consistency score, and quality score, respectively. 4k, 8k, and 16k indicate the required lengths of the generated summary. Scores underlined indicate the highest values within the same experimental setting.

5.2 Consistency Evaluation

Following ProxyQA (Tan et al., 2024), we construct QA pairs to assess whether generated summaries capture key information across papers. The QA set includes *Single-Context* questions answerable from one paper (e.g., "How many samples does Dataset A contain?") and *Cross-Context* questions requiring cross-paper synthesis (e.g., "Compared with Method B, in which tasks does Method A perform better?"). QA pairs are initialized using GPT-4o (Hurst et al., 2024) and refined through iterative LLM and human verification. An initial LLM pass filters for plausibility and usability; human annotators then select valid pairs and deduplicate. If the yield is insufficient, the process repeats—regenerating and re-validating—until the target quantity is met. Each sample contains 6 questions of each type, totaling 1,200 QA pairs.

As the QA pairs are complex and cannot be evaluated by simple matching, we adopt an LLM-as-a-judge framework (Zheng et al., 2023). A judger LLM answers questions based on the generated summary, and an evaluator LLM scores the responses against gold answers on a 0–1 scale. The resulting score defines the Consistency Score S_c:

$$S_c = \frac{\bar{S}_{\text{single}} + \bar{S}_{\text{cross}}}{2}, \tag{6}$$

where $\bar{S}_{\text{single}}$ denotes the average score for all *Single-Context* questions and $\bar{S}_{\text{cross}}$ for all *Cross-Context* questions. Higher values indicate closer alignment with gold answers.

5.3 Quality Evaluation

Beyond content coverage, we evaluate the intrinsic quality of generated summaries, including fluency and structural coherence. Due to the high cost of human evaluation and the limitations of automatic metrics (e.g., perplexity, ROUGE, BLEU), we adopt an LLM-as-a-judge framework following our Consistency Score protocol. We employ a more comprehensive checklist inspired by HelloBench (Que et al., 2024), in which prior experiments have demonstrated that the evaluation metrics align well with human judgments.

In total, we establish $C = 8$ quality aspects, with each aspect comprising $N = 5$ metrics. Detailed information about the quality checklist is presented in Appendix A. The Quality score S_q is then calculated as the average of all metrics across all aspects:

$$S_q = \frac{1}{C}\sum_{i=1}^{C}\left(\frac{1}{N}\sum_{j=1}^{N} S_{i,j}\right), \tag{7}$$

where $S_{i,j}$ is the Quality score of the j-th item in the i-th evaluation aspects.

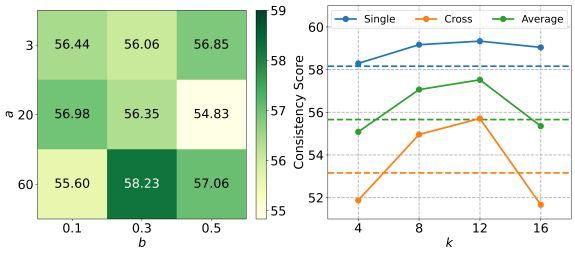
6 Experiments

6.1 Experimental Setting

We evaluated three open-source backbones—Qwen2.5-14B/32B-Instruct (Yang et al., 2024), and LongWriter-glm4-9b (Bai et al., 2025)—using vLLM (Kwon et al., 2023) on NVIDIA A100 (40GB) GPUs. In RAL-WRITER, we adopt bge-base-en-v1.5 (Xiao et al., 2024) for embeddings. For cost efficiency, we use GPT-4o-mini during the QA and evaluation stages, with the temperature set to 0.3 for generation and 0 for evaluation.

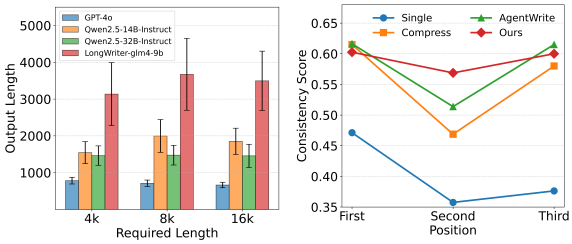
6.2 Baselines

Single The writing instruction is input directly into the LLMs without any preprocessing, and the output generated by the LLM is adopted as the final result. For LongWriter-glm4-9b and GPT-4o, we re-



(a) Heatmap of S_c (b) S_c via k

Figure 6: (a) Heatmap of S_c for various a and b; (b) S_c of RAL-WRITER when k takes different values. "Single" indicates the score on *Single-Context* Questions, "Cross" indicates the score on *Cross-Context* Questions, and "Average" represents the average of the two. The dashed line shows the score of AgentWrite on the corresponding questions.



(a) Length of LLM's output (b) $\bar{S}_{\text{single}}$ via Position.

Figure 7: (a) Output lengths of Single LLMs with varying required lengths. The height of the bars represents the average length of 100 summaries, with error bars indicating the standard deviation across all 100 summaries. (b) $\bar{S}_{\text{single}}$ for Papers at different positions. "Position" refers to the order in which the paper appears in the Input.

stricted experiments to a single round: LongWriter-glm4-9b was specifically fine-tuned for long-text generation tasks, while GPT-4o was limited to this setting due to budget considerations.

AgentWrite (Bai et al., 2025) Similar to RAL-WRITER, this approach employs structured writing planning with sequential paragraph composition but omits the retrieval and restatement mechanisms.

Compress We implement and evaluate the Compress method, which retains AgentWrite's core workflow but introduces a preprocessing stage using LLMLingua model (Jiang et al., 2023) to achieve 50% text compression. To ease the LLM's contextual processing burden, this strategy compresses the input by discarding non-essential content while retaining key information via semantic-aware compression.

6.3 Main Results

Table 2 shows main results of RAL-WRITER and baselines, and we have the following observations:

O1: A model with a larger number of parameters does not necessarily equate to stronger long-context input-output capabilities. Contrary to expectations, Qwen2.5-32B-Instruct does not outperform Qwen2.5-14B-Instruct, and shows slight degradation under both "Single" and "AgentWrite" settings. We attribute this to increased computational overhead under extreme long-context conditions, suggesting that the 14B model may offer better cost-performance trade-offs for long-input-long-output tasks. Future work will examine whether this behavior generalizes across model scales and architectures.

O2: RAL-WRITER enhances the long input and output capabilities of LLMs. The experimental outcomes reveal that RAL-WRITER achieves statistically superior performance in both S_c and S_q metrics compared to baseline approaches. This empirically substantiates that the retrieve-and-restate mechanism effectively enhances knowledge fidelity and linguistic quality in long-form generation tasks. Notably, in length adherence evaluation, RAL-WRITER maintained competitive performance relative to baseline methods, demonstrating the superiority in output regulation while achieving marked improvement when implemented with the Qwen2.5-32B-Instruct architecture.

Step	An overview of Reinforcement learning with human feedback (RLHF), including fitting a reward model and fine-tuning the LM using reinforcement learning.
Top 1	Alignment with Reinforcement Learning: Reinforcement learning with human feedback (RLHF) commonly applies the Bradley-Terry model to..., which combines the reward modeling into the pre-ference learning stage...
Top 2	Aligning generative models with human feedback has been successfully used ... For LLM, align-ment methods such as RLHF have consistently proven to be more beneficial than doing SFT alone... Human feedback is often discussed...
Top 3	At a high level, existing methods instill the desired behaviors into a language model using cu-rated sets of human preferences representing... While RLHF produces models with impressive conversational and coding abilities...

Table 3: The recall of data chunks during the Write phase, showcasing the effectiveness with real data.

O3: LLM Agents continue to face challenges in generating long-form text at the 16k words scale. Even when utilizing the Plan-Write framework for 16k-word generation tasks, consistent length compliance remains unattainable. Through analysis of planning steps, we identified failures in step-wise word count allocation: LLMs occasionally produce planning sequences with insufficient cumulative word count targets, resulting in shorter

Task	AgentWrite	RAL-WRITER	Increase
4k	405.00	535.77	130.77
8k	464.05	672.30	208.25
16k	507.36	674.10	166.74
Overall	458.80	627.39	168.59

Table 4: Average input tokens ($\times 10^3$) per summary at different target lengths.

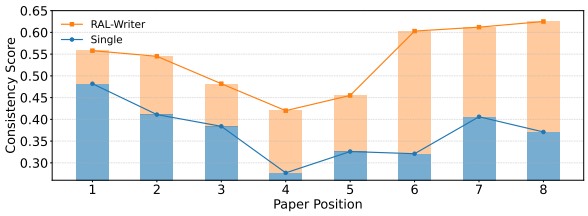


Figure 8: Consistency scores across paper positions, highlighting robustness to the lost-in-the-middle effect.

summaries. This limitation likely stems from inherent reasoning deficiencies in the *Planner* LLM's capability to decompose long-context writing tasks.

6.4 Parameter Analysis

6.4.1 Impact of Parameter a and b

In Eq. (3), parameters a and b control which chunks are affected and to what extent during the retrieval process. We evaluate all 9 combinations of $a \in \{5, 20, 60\}$ and $b \in \{0.1, 0.3, 0.5\}$ using S_c as the selection criterion, as it reflects retrieval quality. Results are shown in Figure 6a. The best configuration ($a = 60$, $b = 0.3$) retrieves semantically relevant chunks aligned with step-specific content requirements, as shown in Table 3.

6.4.2 Impact of Retrieved Chunks Number k

Increasing the number of retrieved chunks (k) improves coverage but also introduces noise, enlarging the input and potentially distracting the model. We therefore tune k by fixing $a = 10$, $b = 0.2$, using Qwen2.5-14B-Instruct with an 8k-word target length, and evaluating $k \in \{4, 8, 12, 16\}$ via the Consistency Score. As shown in Figure 6b, performance peaks at $k = 12$ and drops sharply thereafter, as larger k values approach or exceed the 128k context limit, degrading output quality.

6.5 Analysis of Response Length

Table 2 shows that LLMs generally fail to reach target lengths in single-step generation. Distribution analysis of 100 samples per method (Figure 7a) reveals that GPT-4o rarely exceeds 1,000 words, Qwen2.5 outputs around 2,000 words, and LongWriter-glm4-9b produces mostly under 4,000 tokens despite its long-output capability. Models trained on short-input–long-output tasks perform better on long-input–long-output tasks but still fall short of desired lengths.

6.6 Cost Analysis of Comparable Baselines

Restating input chunks in RAL-WRITER increases input token usage. We compare its overhead with AgentWrite by measuring average total input tokens per summary (Table 4). RAL-WRITER consumes roughly 160k more tokens, which remains manageable compared to other agent-based methods requiring additional invocation rounds.

6.7 The effect of Lost in the Middle

We performed supplementary experiments using 8 research papers on the same topic. Specifically, we generated 8 sequences by cyclically permuting the paper order. For each sequence, we generated a summary using Qwen2.5-14B-Instruct. For each positional slot (1–8), we computed Consistency Scores for the Single-Context Questions of the paper at that position. We then derived average scores for positions 1–8 by aggregating across all permutations. Results are shown in Figure 8.

6.8 Found in the Middle

To further investigate whether RAL-WRITER alleviating the "lost-in-the-middle", we statistically analyzed the Consistency Score in answering *Single-Context* Questions posed in the 1st, 2nd, and 3rd papers, respectively. Corresponding results are shown in Figure 7b. Compared to AgentWrite, Compress, and the single-model invocation, RAL-WRITER exhibited a significantly reduced decline in accuracy when responding to questions related to the 2nd (the middle position) paper, demonstrating that RAL-WRITER is effective in markedly mitigating the "lost-in-the-middle" issue.

7 Conclusion

We introduce RAL-WRITER, a Plan-Write framework with retrieval and restatement, to address the "lost-in-the-middle" problem. We also present a multi-paper summarization dataset with comprehensive evaluations of length, consistency, and quality, facilitating systematic analysis of long-input, long-output generation. This is the first work, to our knowledge, to explicitly investigate long-output generation conditioned on long inputs.

Limitations

We notice that a current long-text generation method involves using a long response corpus for the SFT of LLMs, equipping them with the capability to produce extensive texts. In this context, RAL-WRITER proposed in this paper can generate high-quality long-input and long-output corpora for this SFT. Due to resource constraints, we do not explore further attempts, but we believe this is a highly meaningful direction.

References

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2024. Make your llm fully utilize the context. In *Advances in Neural Information Processing Systems*, volume 37, pages 62160–62188. Curran Associates, Inc.

Merve Astekin, Max Hort, and Leon Moonen. 2024. A comparative study on large language models for log parsing. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 234–244.

Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2025. Longwriter: Unleashing 10,000+ word generation from long context LLMs. In *The Thirteenth International Conference on Learning Representations*.

Harrison Chase. 2022. 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. In *The Twelfth International Conference on Learning Representations*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Jiangnan Fang, Cheng-Tse Liu, Jieun Kim, Yash Bhedaru, Ethan Liu, Nikhil Singh, Nedim Lipka, Puneet Mathur, Nesreen K Ahmed, Franck Dernoncourt, and 1 others. 2024. Multi-llm text summarization. *arXiv preprint arXiv:2412.15487*.

Alexander Gurung and Mirella Lapata. 2025. Learning to reason for long-form story generation. *arXiv preprint arXiv:2503.22828*.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008.

Junqing He, Kunhao Pan, Xiaoqun Dong, Zhuoyang Song, YiBo Liu, Qianguo Sun, Yuxin Liang, Hao Wang, Enming Zhang, and Jiaxing Zhang. 2024. Never lost in the middle: Mastering long-context question answering with position-agnostic decompositional training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13628–13642, Bangkok, Thailand. Association for Computational Linguistics.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Gregory Kamradt. 2023. Needle in a haystack - pressure testing llms.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Philippe Laban, Alexander Fabbri, Caiming Xiong, and Chien-Sheng Wu. 2024. Summary of a haystack: A challenge to long-context LLMs and RAG systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9885–9903, Miami, Florida, USA. Association for Computational Linguistics.

Hao Liu and Pieter Abbeel. 2023. Blockwise parallel transformers for large context models. *Advances in neural information processing systems*, 36:8828–8844.

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

Xiang Liu, Peijie Dong, Xuming Hu, and Xiaowen Chu. 2024b. LongGenBench: Long-context generation benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 865–883, Miami, Florida, USA. Association for Computational Linguistics.

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*.

Chau Minh Pham, Simeng Sun, and Mohit Iyyer. 2024. Suri: Multi-constraint instruction following in long-form text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1722–1753, Miami, Florida, USA. Association for Computational Linguistics.

Bowen Ping, Jiali Zeng, Fandong Meng, Shuo Wang, Jie Zhou, and Shanghang Zhang. 2025. Longdpo: Unlock better long-form generation abilities for llms via critique-augmented stepwise information. *arXiv preprint arXiv:2502.02095*.

Shanghaoran Quan, Tianyi Tang, Bowen Yu, An Yang, Dayiheng Liu, Bofei Gao, Jianhong Tu, Yichang Zhang, Jingren Zhou, and Junyang Lin. 2024. Language models can self-lengthen to generate long texts. *arXiv preprint arXiv:2410.23933*.

Haoran Que, Feiyu Duan, Liqun He, Yutao Mou, Wangchunshu Zhou, Jiaheng Liu, Wenge Rong, Zekun Moore Wang, Jian Yang, Ge Zhang, and 1 others. 2024. Hellobench: Evaluating long text generation capabilities of large language models. *arXiv preprint arXiv:2409.16191*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Haochen Tan, Zhijiang Guo, Zhan Shi, Lu Xu, Zhili Liu, Yunlong Feng, Xiaoguang Li, Yasheng Wang, Lifeng Shang, Qun Liu, and Linqi Song. 2024. ProxyQA: An alternative framework for evaluating long-form text generation with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6806–6827, Bangkok, Thailand. Association for Computational Linguistics.

Tempest A Van Schaik and Brittany Pugh. 2024. A field guide to automatic evaluation of llm-generated summaries. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2832–2836.

Tong Wu, Yanpeng Zhao, and Zilong Zheng. 2024. An efficient recipe for long context extension via middle-focused positional encoding. *Advances in Neural Information Processing Systems*, 37:56349–56373.

Yuhao Wu, Ming Shan Hee, Zhiqiang Hu, and Roy Ka-Wei Lee. 2025. Longgenbench: Benchmarking long-form generation in long context LLMs. In *The Thirteenth International Conference on Learning Representations*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Zhuohan Xie, Trevor Cohn, and Jey Han Lau. 2023. The next chapter: A study of large language models in storytelling. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 323–351.

Ruibin Xiong, Yimeng Chen, Dmitrii Khizbullin, Mingchen Zhuge, and Jürgen Schmidhuber. 2025. Beyond outlining: Heterogeneous recursive planning for adaptive long-form writing with language models. *arXiv preprint arXiv:2503.08275*.

Junjielong Xu, Ruichun Yang, Yintong Huo, Chengyu Zhang, and Pinjia He. 2024. Divlog: Log parsing with prompt enhanced in-context learning. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–12.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Jianxin Yang. 2023. Longqlora: Efficient and effective method to extend context length of large language models. *arXiv preprint arXiv:2311.04879*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024. ∞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.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2024. PoSE: Efficient context window extension of LLMs via positional skip-wise training. In *The Twelfth International Conference on Learning Representations*.

A Checklist Used in Quality Evaluation

Aspects	Metrics
Instruction follow	Originality, citation standards, domain-specific terms usage, format check, comparison dimensions.
Structure analysis	Structural integrity, content logic, transition smoothness, narrative consistency, content redundancy avoidance.
Data utilization	Trend recognition, theory integration, analytical acumen, field contextualization, experimental results accuracy.
Correlation analysis	Methodological differences, complementarity, contribution evaluation, thematic connection, integrative framework.
Insightfulness	Cross-domain impact, application viability, future directions, innovation identification, structure proposal.
Critical thinking	Methodological critique, bias detection, improvement suggestions, findings critique depth, alternative hypothesis proposal.
Reflection	Contribution assessment, method reflection, result generalization, cross-domain connection, testing discussion
Innovation	Innovation analysis, methodological transformation, application creativity, research question impact, cross-domain application innovation.

Table 5: Brief introduction of quality score checklists

B Summary Generation Prompt

You are an experienced researcher, I will give you some scientific research papers in the same field. Please read them carefully and write a summary about them.

Here are the papers:

<paper 1>
{{ paper1 }}
</paper 1>

<paper 2>
{{ paper2 }}
</paper 2>

<paper 3>
{{ paper3 }}
</paper 3>

Your summary should follow these steps:
- Title: Clearly state the main subject or topic of the summary.
- Introduction: Describe the field and briefly introduce its history. Then introduce current progress and challenges.
- Introduce the main content of each paper separately. Then summarize their commonalities and innovations.
- Compare the results of the papers and discuss differences in the results.
- Conclusion: Summarize the main findings and suggest future research directions.

The following are the key points to note:
- If there are important data or main equations in the given papers, remember to mention them in your summary using Markdown.
- Use of tables to compare different approaches is encouraged.
- The first appearance of a professional term must be marked with the full English name and abbreviation.
- Don't directly copy the papers, write the summary in your own words.
- Do not include the titles of reference papers directly in your paper.

Total word count should be about {{ length }} words.

C Writing Steps Planner Prompt

I need you to help me break down the following long-form writing instructions into multiple subtasks. Each subtask will guide the writing of one paragraph in the essay and should include the main points and word count requirements for that paragraph.

The writing instruction is as follows:
<instruction>
{{ instruction }}
</instruction>

Please break it down in the following format, with each subtask taking up one line:
Paragraph 1 - Main Point: [Describe the main point of the paragraph, in detail] - Word Count: [Word count requirement, e.g., 400 words]
Paragraph 2 - Main Point: [Describe the main point of the paragraph, in detail] -

Word Count: [word count requirement, e.g., 1000 words]...

Make sure that each subtask is clear and specific, and that all subtasks cover the entire content of the writing instruction. Do not split the subtasks too finely; each subtask's paragraph should be no less than 200 words and no more than 1000 words. Do not output any other content.

D Retrieve-and-Restate Writer Prompt

You are an excellent writing assistant. I will give you an original writing instruction and my planned writing steps. I will also provide you with the text already written. Please help me continue writing the next paragraph based on the writing instructions, writing steps, and the written text.

Writing instruction:
<instruction>
{{ instruction }}
</instruction>

Writing steps:
<steps>
{{ steps }}
</steps>

Already written text:
<written>
{{ written }}
</written>

I'll restate some parts of the instruction that may need to be used:
<restatement>
{{ restatement }}
</restatement>

Please integrate the original writing instruction, writing steps, and written text, and now continue writing:
<step>
{{ step }}
</step>

Remember to only output the paragraph you write, without repeating the already written text.

E Question-Answer Pair Scoring Prompt

Analyze how well the predicted answer addresses the question based on the standard answer.

<question>
{{ question }}
</question>

<gold>
{{ answer }}
</gold>

<predict>
{{ predict }}
</predict>

Scoring Criteria
- **1.0**: Perfect match - All key points from the standard answer covered with accurate evidence
- **0.75**: Mostly correct - Minor omissions/errors but maintains core understanding
- **0.5**: Partially correct - Addresses > 50 % key elements but misses critical aspects
- **0.25**: Marginally relevant - Only surface-level connection to the question
- **0**: Irrelevant/Incorrect - Contradicts or fails to address the question

Evaluation Steps
1. Cross-check key elements between the standard answer and the predicted answer
2. Verify evidence alignment with reference paper sections
3. Identify:
- Matching components
- Missing critical points
- Additional irrelevant content
- Evidence misinterpretations

Output Format
{
"reason": "Concise analysis comparing

predicted vs standard answer",
"score": "Quantized score (0, 0.25, 0.5, 0.75, 1)"
}

Constraints

- Score MUST reflect discrete tiers (no intermediate values)
- Never reference external knowledge beyond provided inputs
- Maintain strict objectivity in analysis
- Do not output information beyond the specified JSON format

Example Output

{
"reason": "Predicted answer correctly identified the methodology but missed two key limitations mentioned in Conclusion. Added unsupported speculation about applications.",
"score": "0.5"
}