Bridging Cross-Chunk Gaps: A Self-Questioning Approach for Long-Context Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) has been broadly adopted to mitigate hallucinations in large language models (LLMs) by grounding their outputs in external documents. However, when dealing with long, coherently structured texts, the standard assumption that each chunk is self-contained often fails—vital context may span multiple segments. This breakdown undermines retrieval reliability and ultimately impairs generation quality. Our empirical findings reveal that in long-document scenarios, as many as 92% of user queries require cross-chunk semantic dependencies to produce sufficiently supported answers. This observation aligns with cognitive frameworks like the Zeigarnik Effect and Kintsch's Construction-Integration Model, both emphasizing the need to track incomplete information until a coherent whole is formed. To address these challenges, we propose a Self-Questioning RAG (SqRAG) framework. The core idea is to generate and integrate question–answer pairs that explicitly capture inter-chunk connections, thereby enhancing the retrieval process to account for global context rather than isolated segments. Experimental evaluations demonstrate that SqRAG not only reduces hallucinations but also improves coherence and factual accuracy across multiple benchmarks, confirming that modeling cross-chunk dependencies is key to robust and context-rich generation.

1 Introduction

Retrieval-Augmented Generation (RAG) frameworks have emerged as a powerful technique for tasks that involve processing and comprehending long documents. By splitting a lengthy text into manageable chunks, retrieving the most relevant segments, and conditioning a Large Language Model (LLM) on these retrieved passages, RAG systems can address user queries with contextually grounded responses (Gao et al., 2023). However, this approach typically assumes that each retrieved chunk is both sufficiently informative and semantically self-contained (Barnett et al., 2024). In practice, crucial information may be scattered across multiple segments, and simply breaking a document into chunks can disrupt the logical flow and holistic understanding of the text.

Consider the example illustrated in Figure 1 (b), where the user poses the question: "Who is the *federal leader* of the *political party* that *Ken Epp* belongs to?" The document is divided into multiple chunks, each containing crucial pieces of information (e.g., Ken Epp's affiliation with the Conservative Party of Canada and details about its leader). However, a standard RAG pipeline (Lewis et al., 2020) might retrieve only a subset of these chunks—omitting the segment that explicitly identifies the federal leader—thereby producing an incomplete answer (e.g., merely stating Ken Epp's party without mentioning Andrew Scheer). This shortcoming arises from the inability to perform cohesive reasoning across multiple chunks. Although each chunk is accurate in isolation, the system fails to integrate them seamlessly. This exemplifies a critical limitation of RAG for long texts: it lacks an explicit mechanism to preserve semantic continuity and ensure coherence across chunks.

Some works aim to model structured relationships in the retrieval space via knowledge graphs, such as GraphRAG (Edge et al., 2024) and LightRAG (Guo et al., 2024), which organize retrieved text as graphs. GraphRAG and LightRAG construct entity-relation graphs without capturing deeper semantic dependencies across text chunks. Other efforts explore changing the retrieval unit itself (e.g., Dense X Retrieval (Chen et al., 2023)), treating propositions as retrieval units. Nevertheless, as each proposition typically relates to just one discrete chunk, it can be extracted imprecisely or compressed excessively, which neither guarantees accurate semantic representation nor resolves cross-chunk dependencies. Similarly, approaches like

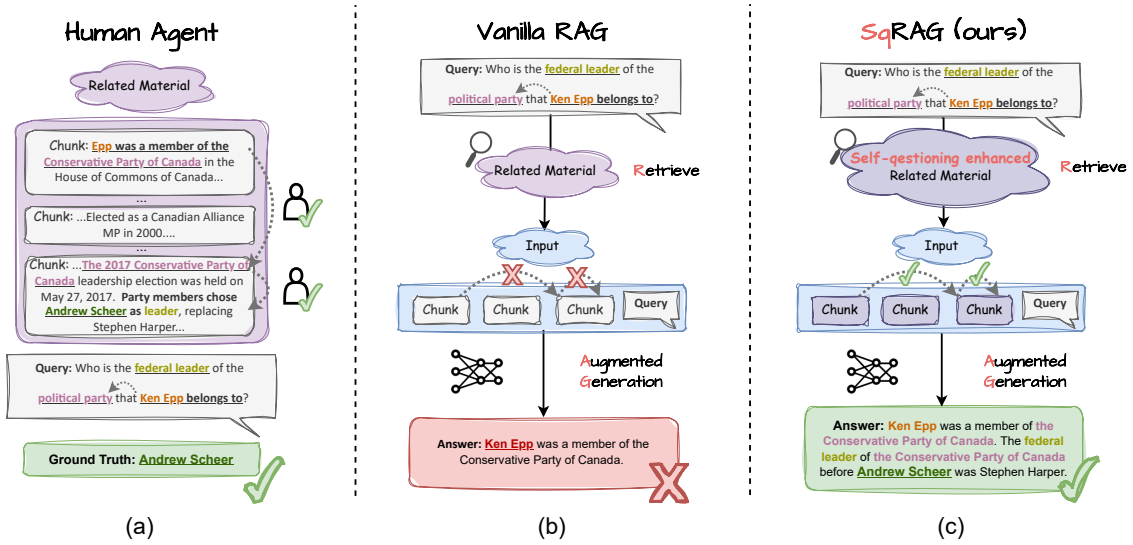

Figure 1: **Comparison of a Human Agent, Vanilla RAG, and Our SqRAG Approach. (a)** A human agent can easily connect information from multiple chunks (e.g., Ken Epp was a member of the Party, and Andrew Scheer is that party's federal leader). **(b)** Vanilla RAG system retrieves only partial information and produces an incomplete answer, indicating Ken Epp's affiliation without identifying the correct federal leader. **(c)** Our SqRAG approach integrates cross-chunk reasoning by prompting the system to ask and answer intermediate questions, leading to a fully grounded response (Andrew Scheer). This example is drawn from one of the cases in our experimental results. For more details, please refer to Table 11 and Table 13 (Appendix E **Case Study**).

FLARE (Jiang et al., 2023) and Self-RAG (Asai et al., 2023) implement active retrieval that allows the LLM to choose what to retrieve on the fly based on user query, but they still rely on discrete text chunks whose semantic content remains fragmented.

A helpful conceptual lens to motivate a more cohesive retrieval strategy is the Zeigarnik Effect, a psychological principle noting that humans tend to remember and maintain focus on incomplete or unresolved tasks (Fox, 2020). By analogy to RAG, one can think of each chunk's semantic dependencies as "unfinished business" that the model should not simply discard upon moving to another segment. By prompting the system to explicitly pose and answer pending questions that arise while reading the document, SqRAG imitates how people hold partial information in mind, awaiting additional context to complete the picture. This process maintains a thread of "semantic suspense" that encourages the model to preserve and revisit cross-chunk connections. Moreover, the question–answer (QA) pairs generated from these dependencies can either serve as standalone retrievable units or be inserted back into the source text as annotations. In both cases, these QA pairs assist the LLM during **R**etrieval, **A**ugmented context construction, and **G**eneration phases, enabling it to produce more coherent and contextually accurate responses, which is shown in Figure 1 (c). Additionally, since these steps occur during the indexing phase, they do not impose extra computational latency at retrieval time (Section 4.4 **Time Cost**).

Our main contributions are as follows:

- We illustrate the ubiquitous presence of cross-chunk semantic dependencies in RAG tasks involving longer texts, and underscore their importance for performance gains. (Section 2)
- We create enhanced documents that better leverages global context for user queries by incorporating cross-chunk semantic information. Compared with RAG methods relying solely on discrete text chunks, our proposed method demonstrates superior performance. (Section 4)
- We analyze the reasons behind the framework's efficacy (Section 4.4 **Case Study**) and further apply the method in reverse, constructing a high-quality Chinese QA dataset for long texts (Section 4.4 **Dataset Generation**).

2 Ubiquity of Cross-Chunk Semantic Dependencies

Semantic dependencies across textual chunks are a pervasive feature of narrative and expository texts,

where crucial information often appears in widely separated segments (Wolfe, 2005; Sangers et al., 2021; Xu et al., 2024). Cognitive theories, such as Kintsch's Construction-Integration Model, suggest that human readers naturally bridge these gaps to construct a coherent mental representation of a text (Soares and Corrêa, 2001). Similarly, the Zeigarnik Effect, introduced in the *Introduction* section, underscores how unresolved dependencies in earlier segments can shape and enhance comprehension when later information emerges. These perspectives collectively highlight the foundational role that cross-chunk dependencies play in generating contextually grounded and cohesive reasoning.

To empirically examine the influence of cross-chunk semantic dependencies, we conducted an experiment on the NarrativeQA (Kočiskỳ et al., 2018) dataset, obtained from the LongBench (Bai et al., 2024) framework. This dataset consists of long-form narratives intended to test the ability of models to synthesize context from distributed textual segments. Specifically, each narrative was split into 512 token chunks, and for each chunk, GPT-4o-mini (Hurst et al., 2024) was used to generate ten contextually relevant questions. We then combined all questions from every chunk to form a single, pooled question set. Another GPT-4o-mini was tasked with answering a subset of 40 semantically similar questions for each chunk, thereby simulating both retrieval and multi-chunk reasoning. Our metrics encompassed the total number of questions answered, the percentage of valid answers, and the proportion of valid answers that drew on intra-chunk[1] versus cross-chunk[2] information.

Figure 2 presents the results of this experiment. Remarkably, 92% of the valid answers required integrating content across multiple chunks, whereas only 8% of valid answers drew solely on a single chunk. This finding strongly indicate that methods treating each chunk as self-contained are insufficient for effectively handling long documents. Instead, a strategy that explicitly models and reconciles dependencies spanning multiple chunks is essential. By capturing and integrating cross-chunk semantic dependencies, RAG frameworks can generate answers that are not only more accurate but also more coherent and contextually aligned. Such an approach leverages question-driven semantic linking to establish a cohesive retrieval mechanism—a strategy at the core of SqRAG, which we detail in the following sections.

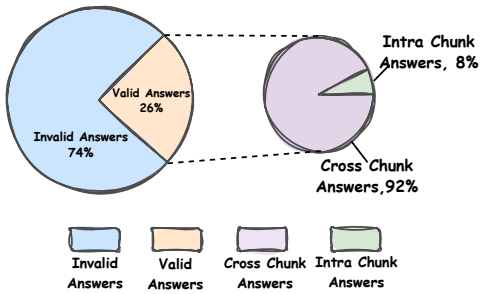

Figure 2: **Why Do We Need Cross-Chunk Information?** According to the pie chart, 92% of valid answers depend on information from multiple chunks (i.e., cross-chunk answers), while only 8% stem from a single chunk (i.e., intra-chunk answers).

3 SqRAG: Self-Questioning RAG System

3.1 Overview

Building on the insight that cross-chunk dependencies are crucial for long-text question answering, our method (depicted in Figure 3) is divided into two main stages: an **Index Phase** (Algorithm 1) and an **Inference Phase** (Algorithm 2). In the **Index Phase**, the original document $\mathcal{D}$ is split into chunks $\mathcal{N} = \{n_1, n_2, \ldots, n_i\}$. We then use a dedicated Q-LLM to generate questions for each chunk, gather these questions to a question database QuestionsDB and employ an A-LLM to produce corresponding answers for relevant questions in this QuestionsDB. All validated question–answer pairs are merged into a central repository, qaDB. In the **Inference Phase**, we retrieve from qaDB the QA pairs most relevant to the user query q_u. These pairs are then inserted back into their respective chunks to form an enhanced document $\mathcal{N}_{\text{enh}}$, which incorporates sufficient cross-chunk dependencies. Finally, we adopt a retrieval mechanism combined with an ANSWER-LLM—essentially a RAG-based process where the retrieval is performed over $\mathcal{N}_{\text{enh}}$—to generate the refined response a_u. Further details about the notation and functions used throughout, are provided in Appendix A.

[1]The chunk containing the question is the same as the chunk providing the valid answer.

[2]The chunk containing the question differs from the chunk providing the valid answer.

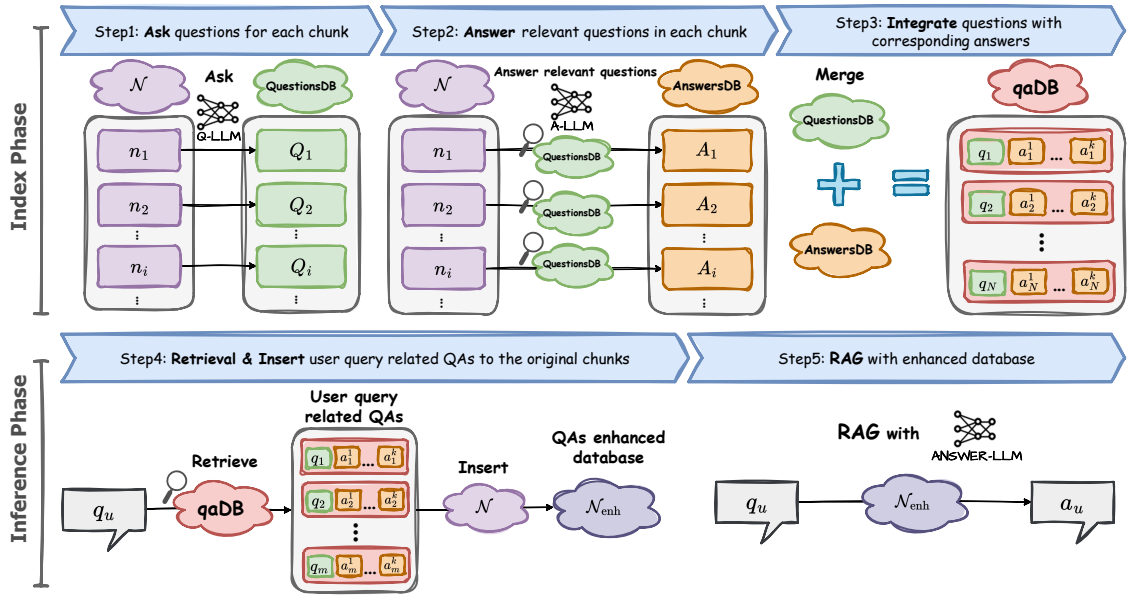

Figure 3: Overview of the Self-Questioning RAG Method. Table 11 presents real examples of each data structure used in the method from an actual experiment.

3.2 Index Phase

During the **Index Phase**, the user uploads a document-based knowledge repository, which the system converts into a searchable index. During this step, we extract question–answer pairs from the original document, explicitly capturing any cross-chunk dependencies.

Step 0: Chunk Splitting. We first invoke the function $\text{Split}(\mathcal{D}, c_s)$, which partitions the document $\mathcal{D}$ into overlapping segments of size c_s. This yields a set of chunks, $\mathcal{N} = \{n_1, n_2, \ldots, n_i, \ldots\}$.

Step 1: Question Generation. Next, for each chunk $n_i \in \mathcal{N}$, we use Q-LLM to generate potential questions by invoking $Q_i = \text{Q-LLM}(n_i)$. It is to capture any inquiries that may arise from the local context of each chunk.

Step 2: Answer Generation. For each chunk n_i, a relevant subset of questions Q_i' is selected from QuestionsDB. These relevant questions serve as "semantic placeholders" for bridging different chunks of the document. The function $\text{SelectRelevant}(\text{QuestionsDB}, n_i)$ treats each question in QuestionsDB as a retrieval unit, using content of n_i as the query. Corresponding answers and their evidences[3] A_i are then generated by $\text{A-LLM}(Q_i', n_i)$. This step enables the model to articulate the knowledge required to answer questions that may span both local and broader contexts within the text. To ensure answer quality, the function $\text{Eval}(A_i)$ filters out incomplete or incorrect responses by detecting a predefined prefix, "[I cannot answer]," which is accomplished by prompting A-LLM. Only validated answers are retained in AnswersDB.

Step 3: Integrate Questions and Answers. Finally, we combine QuestionsDB and AnswersDB into a unified repository, resulting in a cohesive database of question–answer pairs that reference specific chunks in $\mathcal{N}$. By this design, qaDB captures cross-chunk dependencies in these pairs, thereby streamlining retrieval in the subsequent Inference Phase. Concretely, Merge pairs each question q_i from QuestionsDB with its corresponding valid answers and evidence a_i stored in AnswersDB (linked via pre-saved ask-node and answered-node IDs). This approach explicitly models cross-chunk dependencies by associating questions and answers across different chunks.

3.3 Inference Phase

In the **Inference Phase**, the user poses a query to the system, which then leverages the previously uploaded knowledge base to assist in generating answers. Specifically, we insert query-relevant question–answer pairs back into the original document to create an enhanced version, and we subsequently apply a standard RAG process on this enhanced document.

[3]Evidence refers to the original text that supports the answer, with a one-to-one correspondence to each answer.

Algorithm 1 SqRAG in Index Phase

Input: $\mathcal{D}$: support document, c_s: chunk size, Q-LLM: question LLM , A-LLM: answer LLM, Eval: answer evaluation method
Step 0: Chunk Splitting.
$\mathcal{N} \leftarrow \text{Split}(\mathcal{D}, c_s)$

Step 1: Question Generation
for each $n_i \in \mathcal{N}$ **do**
  $Q_i \leftarrow \text{Q-LLM}(n_i)$
  $\text{QuestionsDB} \leftarrow \text{QuestionsDB} \cup Q_i$
end for

Step 2: Answer Generation
for each $n_i \in \mathcal{N}$ **do**
  $Q_i' \leftarrow \text{SelectRelevant}(\text{QuestionsDB}, n_i)$
  $A_i \leftarrow \text{A-LLM}(Q_i', n_i)$
  $A_i' \leftarrow \text{Eval}(A_i)$
  $\text{AnswersDB} \leftarrow \text{AnswersDB} \cup A_i'$
end for

Step 3: Integrate Questions & Answers
$\text{qaDB} \leftarrow \text{Merge}(\text{QuestionsDB}, \text{AnswersDB})$
Output: questions with answers database qaDB

Step 4: Retrieve & Insert User Query Relevant QAs. Given a user query q_u, we first identify a subset of question–answer pairs relevant to q_u by calling $\text{SelectRelevant}(\text{qaDB}, q_u)$. The function $\text{SelectRelevant}(\text{qaDB}, q_u)$ treats each question in qaDB as a retrieval unit, using the content of q_u as the retrieval query. After retrieval, we apply $\mathcal{N}_{\text{enh}} = \text{Insert}(\mathcal{N}, \text{RelevantQAs})$, augmenting document $\mathcal{N}$ with the user query relevant QA pairs RelevantQAs. The content of each QA insertion is formatted as "({question q_i} {answer a_i^k})"[4], similar to a note taken while reading a book. The insertion location is determined based on fuzzy string matching between the chunk content and the QA pair content. A buffer count is used during the insertion process to ensure there is no QA overlap. This insertion process effectively integrates the previously generated cross-chunk insights into the primary text body. After this step, we obtain an enhanced document $\mathcal{N}_{\text{enh}}$ with the same format as the original document $\mathcal{N}$ but enriched with thoughtful, cross-dependent notes embedded within.

Step 5: RAG with Enhanced Database. Lastly, the system performs a standard RAG pipeline on the enhanced document $\mathcal{N}_{\text{enh}}$. It begins by retrieve $\mathcal{R} = \text{Retrieve}(\mathcal{N}_{\text{enh}}, q_u)$, and then augmented generates a final response $a_u = \text{ANSWER-LLM}(q_u, \mathcal{R})$. In practice, this approach yields more accurate and contextually grounded responses compared to performing RAG on the original document. The effectiveness of leveraging question-driven links will be discussed in detail in the following experimental results.

[4]For example, "(Who is Charlie? Charlie is my pet.)".

Algorithm 2 SqRAG in Inference Phase

Input: qaDB, $\mathcal{N}$: chunks of document, q_u: user query, ANSWER-LLM: LLM to answer user query
Step 4: Retrieve & Insert User Query Relevant QAs
$\text{RelevantQAs} \leftarrow \text{SelectRelevant}(\text{qaDB}, q_u)$
$\mathcal{N}_{\text{enh}} \leftarrow \text{Insert}(\mathcal{N}, \text{RelevantQAs})$

Step 5: RAG with Enhanced Database
$\mathcal{R} \leftarrow \text{Retrieve}(\mathcal{N}_{\text{enh}}, q_u)$
$a_u \leftarrow \text{ANSWER-LLM}(q_u, \mathcal{R})$
Output: LLM response a_u

4 Experiment

4.1 Experiment Setup

Datasets and Models. For our experiments, we use four datasets from LongBench (Bai et al., 2024): NarrativeQA (Kočiskỳ et al., 2018), MuSiQue (Trivedi et al., 2022), TriviaQA (Joshi et al., 2017) and HotpotQA (Yang et al., 2018). These datasets are carefully chosen to cover diverse tasks such as single-document long context QA (NarrativeQA), multi-document QA (HotpotQA and MuSiQue), and few-shot QA (TriviaQA), enabling a comprehensive evaluation of long-context understanding. We use two types of Large Language Models (LLM) as our ANSWER-LLM, namely open source LLM (e.g. Llama 3.1 8B (Dubey et al., 2024)) and close source LLM (e.g. GPT-4o-mini (Hurst et al., 2024)). For more details, please refer to Appendix B.1.

Implementation Details. As summarized in Table 4, we compare three baseline methods that exemplify distinct categories of RAG techniques: Vanilla RAG (Lewis et al., 2020), Dense X Retrieval (Chen et al., 2023), and FLARE (Jiang et al., 2023). Vanilla RAG belongs to the "Basic RAG" category, constituting the canonical retrieve-and-

Method	Overall Metrics (close source LLM)			Overall Metrics (open source LLM)		
	Precision	Recall	F1	Precision	Recall	F1
#NarrativeQA#						
Vanilla RAG	29.55	46.20	27.07	29.00	41.58	26.07
Dense X Retrieval	28.38	35.52	23.05	29.25	38.38	25.72
FLARE	25.75	**47.98**	25.45	23.62	20.02	13.25
SqRAG	**31.70**	45.72	**28.28**	**31.30**	**42.80**	**27.15**
#MuSiQue#						
Vanilla RAG	31.35	40.60	26.70	27.92	32.00	22.97
Dense X Retrieval	31.90	39.25	26.78	29.57	32.55	23.25
FLARE	29.38	**47.42**	27.70	25.85	29.72	18.98
SqRAG	**32.92**	41.95	**28.35**	**32.43**	**33.48**	**25.68**
#TriviaQA#						
Vanilla RAG	**90.68**	71.12	73.90	81.85	**71.10**	69.70
Dense X Retrieval	90.03	70.47	72.80	80.53	69.15	67.22
FLARE	77.73	67.72	65.12	74.93	70.88	64.28
SqRAG	90.43	**71.62**	**74.47**	**82.57**	70.82	**70.12**
#HotpotQA#						
Vanilla RAG	48.78	60.85	45.98	43.40	51.37	39.62
Dense X Retrieval	**49.38**	**61.58**	**46.17**	42.85	50.48	38.05
FLARE	37.10	60.65	36.50	35.70	38.60	26.57
SqRAG	48.92	60.50	45.78	**45.85**	**51.50**	**41.12**

Table 1: Main results.

generate pipeline and serving as a foundational reference point for subsequent enhancements. Dense X Retrieval represents "RAG with Modified Retrieval Units," employing a dense embedding-based retrieval process that refines retrieval precision and boosts generation quality. FLARE falls under the "Active Retrieval RAG" category, introducing comprehensive optimizations at the interface of retrieval and generation to reinforce factual consistency and interactive elements. Because knowledge-graph-based baselines require excessive computational resources, we do not include them for comparison in this work.

We implement all baseline methods and our own approach using the LlamaIndex® framework, along with prompt engineering to optimize the interaction between the retriever and the generator. Detailed prompt templates and strategies can be found in Appendix B.2.

Evaluation Metrics. To evaluate the effectiveness of SqRAG, we use **RAGCHECKER** (Ru et al., 2024), which provides fine-grained claim-level analysis. We focus on overall system performance and do not utilize its diagnostic retriever or generator metrics, as comparing different retriever and generator capabilities is not our primary goal. To reduce randomness in the evaluation, we employ four open-source LLMs (i.e., Llama 3.1 8B (Dubey et al., 2024), Qwen 2.5 7B (Yang et al., 2024), Mixtral 8x7B (Jiang et al., 2024), Gemma 2 9B (Team et al., 2024)) as evaluators and ensemble their scores to form the final outcome. For more details, please refer to Appendix B.3.

4.2 Main Results

Table 1 summarizes the main outcomes for our method and baselines (Vanilla RAG, Dense X Retrieval, and FLARE) under both *closed-source* and *open-source* LLM evaluations. Across NarrativeQA and MuSiQue, SqRAG generally achieves the highest F1 scores, affirming the advantage of explicit cross-chunk QA pairs. On TriviaQA, we also outperform other baselines, although the overall gap is less pronounced because TriviaQA questions often require relatively less complex cross-chunk reasoning. Notably, FLARE sometimes attains higher recall, as it tends to generate overly long or verbose outputs, thereby increasing recall at the expense of precision. For HotpotQA—which inherently involves shorter text spans—our method does not always excel in the closed-source setting; however, it demonstrates a clear advantage in the open-source scenario, indicating that more limited retrieval capabilities or LLM resources can amplify the benefits of our cross-chunk linking strategy.

4.3 Ablation Study

Table 2 presents ablation results on the NarrativeQA dataset, highlighting the contribution of each component in our SqRAG framework. During the **Index Phase**, removing the questions relevant selection step (*No* SelectRelevant(QuestionsDB, n_i)) slightly degrades the F1 score to 27.10, indicating that focusing on chunk-relevant questions is crucial. Eliminating the evaluation function (*Remove* Eval(A_i)) also lowers the final performance, underscoring the importance of filtering out invalid or low-quality answers before populating the QA database.

METHOD	PRECISION	RECALL	F1
SqRAG	**31.70**	**45.72**	**28.28**
#Index Phase#			
No SelectRelevant(QuestionsDB, n_i)	30.20	44.50	27.10
Remove Eval(A_i)	30.40	44.17	27.30
#Inference Phase#			
No RelevantQAs	28.25	43.88	24.95
Vanilla RAG	29.55	46.20	27.07
No Retrieval	20.50	31.32	17.45

Table 2: Ablation Study on NarrativeQA Dataset.

During the **Inference Phase**, excluding the insertion of relevant QA pairs (*No* RelevantQAs) yields a more substantial drop to 24.95 F1, suggesting that injecting contextually linked QA pairs effectively facilitates cross-chunk reasoning. Comparisons with Vanilla RAG (27.07 F1) and a scenario where retrieval is omitted entirely (*No Retrieval*, 17.45 F1) further confirm that both retrieval and question-driven augmentation are essential. Overall, these results demonstrate that each mechanism—including question relevance selection, answer evaluation, and relevant QA insertion—plays a critical role in improving long-text QA performance.

4.4 Discussion

Time Cost. Table 3 compares the runtime of SqRAG against Vanilla RAG during the inference phase. SqRAG introduces two additional stages: SelectRelevant(qaDB, q_u) and Insert. As shown in Table 3, these operations incur only negligible overhead for both closed-source (GPT-4o-mini) and open-source (Llama 3.1 8B) LLM settings.

Answers Numbers. An important factor in SqRAG is the quantity of generated answers (*answer numbers*) inserted into each chunk. Small values (e.g., $aq = 10$) may fail to provide sufficient searchable context, while excessively large values (e.g., $aq = 60$ or $aq = 80$) risk introducing substantial noise. Moderate values (e.g., $aq = 20$ or $aq = 40$) appear to strike a balance between additional searchable context and minimal noise, yielding stronger overall F1. For a more comprehensive analysis of answer selection strategies and numerical results, please refer to Appendix C.

	STAGE	CLOSE LLM	OPEN LLM
SqRAG	SelectRelevant(qaDB, q_u)	0.0077	0.0089
	$\mathcal{N}_{\text{enh}} \leftarrow$ Insert($\mathcal{N}$, RelevantQAs)	0.0039	0.0022
	RAG with Enhanced Database	0.3788	0.0250
Vanilla RAG	RAG with Original Database	0.3696	0.0231

Table 3: Comparison of time costs (in seconds) between SqRAG and Vanilla RAG during the inference phase.

From Local to Global. Table 10 illustrates how varying chunk sizes (256, 512, and 1024 tokens) influence question-answering (QA) performance by balancing local detail and global coverage. Smaller chunks offer fine-grained local context, while larger chunks provide a more holistic overview. For a more detailed discussion of these trade-offs, please refer to Appendix D.

Case Study. We present a case study to demonstrate how different RAG methods can exhibit distinct error patterns and successes when handling long-text QA tasks. By examining concrete examples of system outputs, readers can gain a more nuanced understanding of each model's strengths, potential pitfalls, and the types of errors that may arise. Details can be seen in Appendix E.

Dataset Generation. Beyond serving as a retrieval augmentation technique, our method can also facilitate the construction of high-quality QA datasets. In particular, since our framework focuses on identifying and explicitly modeling cross-chunk semantic dependencies, the question–answer pairs it produces tend to capture more complex reasoning steps that traditional RAG methods struggle to retrieve and combine accurately. We ran SqRAG on longer Chinese classical novels and manually filtered out QA pairs with abundant cross-chunk dependencies. These QA pairs naturally spotlight intricate textual relationships—where critical information is scattered across multiple segments—thereby ensuring that the resulting dataset challenges models to incorporate global context and multi-step reasoning. This can be especially valuable for developing new benchmarks or refining existing ones, pushing QA systems beyond simple fact extraction into deeper comprehension and inference. Furthermore, compared with traditional methods of generating QA datasets, our method

Category	Representative Methods	Characteristics	Shortcome
Basic RAG	Vanilla RAG (Lewis et al., 2020)	Splits text into coarse blocks and retrieves a subset.	It struggles in long-document scenarios lacking broader context.
RAG with Modified Retrieval Units	Dense X Retrieval (Chen et al., 2023), HiQA (Chen et al., 2024)	Converts documents into modified retrieval units.	It may overlook semantic dependencies that span multiple propositions.
Knowledge Relationship or Graph Based RAG	GraphRAG (Edge et al., 2024), LightRAG (Guo et al., 2024), HybGRAG (Lee et al., 2024)	Builds knowledge relationships or graphs from documents.	These capture entity relationships but may lose rich contextual cues beyond the graph.
Active Retrieval RAG	FLARE (Jiang et al., 2023), Self-RAG (Asai et al., 2023), Adaptive-RAG (Jeong et al., 2024), CRAG (Yan et al., 2024), CtrlA (Liu et al., 2024), Astute RAG (Wang et al., 2024a), Speculative RAG (Wang et al., 2024b), Agentic RAG (Singh et al., 2025)	Dynamically selects refines retrieved content.	It can lead to lengthy outputs and often still uses naive text chunks lacking cross-block semantic linkage.

Table 4: Summary of RAG methods

can effectively select questions containing cross-chunk information, thereby greatly reducing the burden of constructing a high-quality QA dataset, whereas previous approaches often struggled to produce questions of such caliber and required significant manual effort to sift through numerous generated questions. We showcase representative examples of these generated QA pairs in Appendix F, illustrating how they encapsulate nuanced, cross-chunk relations not readily evident within standard, locally-focused question sets.

5 Related Work

Vanilla RAG (Lewis et al., 2020) combines a pre-trained language model and an external knowledge source by first retrieving relevant documents and then generating answers. However, it may fail to capture cross-chunk semantic dependencies when dealing with long documents.

RAG with modified retrieval units employs smaller or more fine-grained text segments. Dense X Retrieval (Chen et al., 2023) highlights that retrieval granularity (e.g. propositions) can boost performance in certain tasks, while HiQA (Chen et al., 2024) refines multi-document retrieval for large-scale question answering. Both approaches address the rigid chunking of standard RAG but risk overlooking cross-segment semantics.

Knowledge relationship based RAG methods build entity-level relationship or graphs to structure and retrieve relevant contexts. GraphRAG (Edge et al., 2024) extracts a graph of key entities for query-focused summarization, LightRAG (Guo et al., 2024) simplifies index construction for efficient retrieval, and HybGRAG (Lee et al., 2024) fuses textual and relational knowledge. These approaches capture entity relationships but may lose rich contextual cues beyond the graph.

Active retrieval RAG techniques iteratively refine retrieval and generation. FLARE (Jiang et al., 2023) and Self-RAG (Asai et al., 2023) actively query external sources when low-confidence content is detected, while Adaptive-RAG (Jeong et al., 2024) adjusts its strategy based on question complexity. CRAG (Yan et al., 2024) and Speculative RAG (Wang et al., 2024b) introduce corrective or drafting steps to improve factual grounding, CtrlA (Liu et al., 2024) leverages internal probes for guided retrieval, and Astute RAG (Wang et al., 2024a) mitigates imperfect augmentation and knowledge conflicts. Agentic RAG (Singh et al., 2025) surveys these agent-like approaches, illustrating their potential to adapt retrieval behavior on the fly, but they often still rely on naive chunked text lacking deeper semantic linkages.

Table 4 summarizes these RAG methods.

6 Conclusion

In this paper, we highlighted a key RAG challenge: cross-chunk dependencies within lengthy, coherent texts. Our empirical findings show that most real-world questions span multiple segments, often overlooked by conventional RAG pipelines. Motivated by psychological theories, we introduced SqRAG, a framework that systematically generates, integrates QA pairs to bridge these gaps and enrich contextual links. Experiments across diverse datasets reveal that SqRAG outperforms existing approaches in precision, recall, and overall coherence. We further demonstrated how tuning the number of QA pairs, adjusting chunk sizes can refine results. We hope this work inspires deeper exploration of cross-chunk semantics for more robust and context-aware retrieval generation systems.

Limitations

Although SqRAG enhances retrieval robustness and reduces hallucinations, it has practical constraints. The *index phase* can be computationally

expensive, especially for large or frequently updated corpora. Moreover, SqRAG relies on having sufficiently capable LLMs to handle enriched context and cross-chunk cues. Less powerful models may not realize the same gains, limiting the method's overall impact. Consequently, there is a trade-off between the computational overhead required and the precision benefits achieved for long-text retrieval and generation.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. 2024. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. *arXiv preprint arXiv:2412.15204*.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven failure points when engineering a retrieval augmented generation system. *2024 IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, pages 194–199.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2023. Dense x retrieval: What retrieval granularity should we use? *arXiv preprint arXiv:2312.06648*.

Xinyue Chen, Pengyu Gao, Jiangjiang Song, and Xiaoyang Tan. 2024. Hiqa: A hierarchical contextual augmentation rag for massive documents qa. *arXiv preprint arXiv:2402.01767*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Jeremy G. Fox. 2020. Recovery, interrupted: The zeigarnik effect in emdr therapy and the adaptive information processing model. *Journal of EMDR Practice and Research*, 14:175 – 185.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *ArXiv*, abs/2312.10997.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

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

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. corr abs/2403.14403 (2024). *CoRR*, 2403.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N Ioannidis, Huzefa Rangwala, and Christos Faloutsos. 2024. Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases. *arXiv preprint arXiv:2412.16311*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Huanshuo Liu, Hao Zhang, Zhijiang Guo, Kuicai Dong, Xiangyang Li, Yi Quan Lee, Cong Zhang, and Yong Liu. 2024. Ctrla: Adaptive retrieval-augmented generation via probe-guided control. *arXiv preprint arXiv:2405.18727*.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, et al. 2024. Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Nina L Sangers, Jacqueline Evers-Vermeul, Ted JM Sanders, and Hans Hoeken. 2021. Narrative elements in expository texts: A corpus study of educational textbooks. *Dialogue & Discourse*, 12(2):115–144.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*.

Adriana Benevides Soares and Carla Patrícia Quintanilha Corrêa. 2001. Learning and memory: A cognitive approach about the role of memory in text comprehension.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Fei Wang, Xingchen Wan, Ruoxi Sun, Jiefeng Chen, and Sercan Ö Arık. 2024a. Astute rag: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models. *arXiv preprint arXiv:2410.07176*.

Zilong Wang, Zifeng Wang, Long T Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, et al. 2024b. Speculative rag: Enhancing retrieval augmented generation through drafting. *CoRR*.

Michael BW Wolfe. 2005. Memory for narrative and expository text: independent influences of semantic associations and text organization. *Journal of Experimental Psychology: Learning, memory, and cognition*, 31(2):359.

Liyan Xu, Jiangnan Li, Mo Yu, and Jie Zhou. 2024. Fine-grained modeling of narrative context: A coherence perspective via retrospective questions. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5822–5838.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

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

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

A Notation and Functions

Table 5 and Table 6 present the symbol and function descriptions used in Algorithm 1 and Algorithm 2, respectively.

B Experiment Setup

B.1 Datasets and Models details

LongBench is a bilingual, multitask benchmark designed to assess the long-context understanding capabilities of large language models. It consists of 21 tasks across six categories: single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion. In English, the dataset contains an average context length of 6,711 words, whereas in Chinese it reaches 13,386 characters, providing a comprehensive framework for evaluating long-text applications. Due to its focus on extended textual contexts, its broad task coverage, and its authoritative curation by reputable institutions, we consider it a thorough assessment of models' performance in diverse and realistic long-text understanding scenarios. In our study, we employ four datasets from LongBench: NarrativeQA, MuSiQue, TriviaQA, and HotpotQA. Table 7 summarizes their key features.

In terms of large language models, we utilize the lightweight and representative GPT-4o-mini and Llama 3.1 8B as the closed-source and open-source LLMs, respectively, to ensure a comprehensive evaluation of the experimental results.

B.2 Prompts

Figure 4 is the prompt for Q-LLM. Figure 5 is the prompt for A-LLM.

B.3 Evaluation Metrics details

To thoroughly evaluate our RAG system, we integrated **RAGCHECKER** into our framework, which is a fine-grained evaluation framework designed to assess both the retrieval and generation components of RAG systems. It offers a suite of diagnostic metrics that provide detailed insights into system performance. By extracting factual claims from generated responses and validating them against ground-truth answers, RAGCHECKER enables precise evaluations of correctness and completeness.

We implemented RAGCHECKER using the LlamaIndex® framework, which offers seamless integration and robust support for its functionalities. To streamline model inference and manage computational resources efficiently, we employed Ollama® as a proxy server for model execution.

To reduce potential biases and randomness in our evaluation process, we employed four open-source LLMs as evaluators. Each model independently assessed the generated responses, assigning individual scores based on RAGCHECKER's metrics. We then averaged these scores to yield the final evaluation outcome, ensuring a balanced and comprehensive assessment. Table 8 provides an example of these evaluation scores.

C Answer Numbers

Table 9 illustrates how varying the number of answers inserted into each chunk (*aq*) influences the overall performance of our method on the NarrativeQA dataset. A smaller value of *aq* (e.g., $aq = 10$) provides insufficient additional context to improve retrieval recall and F1 scores. Conversely, as *aq* grows (e.g., $aq = 60$ or $aq = 80$), although recall can improve in certain settings, the risk of overloading each chunk with noisy QA pairs rises, ultimately lowering precision and diminishing the final F1. Thus, there is a trade-off between capturing enough information to enhance retrieval and overpopulating the chunks with less relevant details. Our experiments suggest that moderate *aq* values (e.g., $aq = 20$ or $aq = 40$) often yield an optimal balance of informativeness and precision, resulting in more consistent gains across both closed-source and open-source LLM evaluations.

D From Local to Global

Table 10 illustrates how altering the chunk size (e.g., 256, 512, and 1024 tokens) shifts the balance between local context and global coverage. Smaller chunks (e.g., 256 tokens) provide fine-grained local detail but tend to increase the total number of chunks, potentially yielding more local QA pairs while making it harder to build a holistic view of the document. Conversely, larger chunks (e.g., 1024 tokens) capture broader context in fewer segments, thereby reducing the number of available QA pairs and possibly losing nuanced local information.

Our results show that medium-sized chunks (e.g., 512 tokens) often strike a desirable trade-off: they retain enough local structure to generate meaningful QA pairs while also offering sufficient global context for robust cross-chunk reasoning. Notably,

although larger chunks can sometimes improve recall (e.g., FLARE with 1024 tokens), precision may suffer if the model struggles to focus on the most relevant information. Overall, an intermediate chunk size appears most conducive to effectively leveraging cross-chunk dependencies for long-text QA tasks.

E Case Study

E.1 Overview

In this section, we present a detailed case study of erroneous answers produced by several RAG-based models. To facilitate a structured analysis, we categorize these mistakes into four main error types and summarize in Table 12. By breaking down the types of errors and looking at cases from multiple datasets and multiple baselines, we can more intuitively evaluate different responses and analyze the causes of errors in detail.

E.2 Cases of Data Structure in Method

In this section, we present a detailed case study of the data structure used in our method, as shown in Table 11. As explained in section 3, the questions are generated based on each chunk and stored in QuesionsDB (**Step 1**), and answers are generated by A-LLM and stored in AnswersDB (**Step 2**). Each question may corresponds to several answers from different chunks. The qaDB is formed by these question-answer pairs, which contain sufficient cross-chunk information for further RAG process (**Step 3**). The colors in Table 11 represent relevant information that are demand for giving true response to the query. Our method retrieves the question-answer pair that related to the query according to these information and insert them into the original context (**Step 4**). The question-answer pairs would provide sufficient cross-chunk evidence along with the original content to the RAG process. SqRAG then retrieves the top-k chunks to generate a more complete final answer (**Step 5**). The ground truth and the result of Vanilla RAG and our method are shown in the bottom of the table. As shown in the right part of Figure 1, the question-answer pairs in brackets of the retrieved chunks contains cross-chunk evidence and these information helps the model to integrate relevant information and logical relationships between chunks and generate a more complete answer, while the Vanilla RAG simply focusing on each individual chunk to response.

E.3 Cases of Different Methods' responses

Table 13, Table 14, Table 15, Table 16, Table 17, Table 18, Table 19, Table 20, Table 21 are cases of different methods' responses. We categorize these responses into types in Table 12 and compare the results. These cases highlight common error patterns in retrieval-augmented generation and demonstrate how *SqRAG* mitigates them by effectively inserting cross-chunk question-answer pairs (Section 3). The additional retrieved Q&A content better pinpoints relevant context for the final generation step, yielding more accurate and complete responses.

F Dataset Generation

Figure 6 visually compares the average document length of our newly generated dataset, LONGNOVELQA, to several established long-text QA benchmarks, including L-Eval, TriviaQA, HotpotQA, MuSiQue, NarrativeQA, and BookCorpus. Notably, as shown in the figure, most existing datasets have average lengths ranging from roughly 7,000 tokens (L-Eval) up to around 89,000 tokens (BookCorpus). In contrast, our LONGNOVELQA corpus consists of documents exceeding **600,000 tokens** on average, underscoring a substantial increase in text length that encourages deeper cross-chunk reasoning, multi-step inference, and long-context understanding.

Table 22 further illustrates how SqRAG leverages this vast context to generate question–answer pairs that span multiple segments of text, highlighting intricate dependencies and rendering purely local retrieval strategies inadequate. In many cases, questions demand synthesizing information from distant parts of a text, compelling the model to integrate clues that may be scattered across hundreds of pages. This extensive context also paves the way for more complex inference, going beyond straightforward entity matching to reveal whether a model can handle deeper reasoning. Additionally, rather than relying on single-sentence queries or answers, our dataset features longer, interlinked passages, demonstrating how multiple pieces of text must be retrieved, fused, and logically connected to arrive at a correct solution. Together, these characteristics underscore the dataset's emphasis on robust cross-chunk integration and genuinely challenging long-text QA.

SYMBOL	DESCRIPTION
$\mathcal{D}$	The original support document.
c_s	Chunk size parameter for splitting the document.
$\mathcal{N}$	A set of document chunks (nodes) obtained by splitting $\mathcal{D}$.
n_i	Each $n_i \in \mathcal{N}$ is one chunk.
Q_i	A set of candidate questions generated by Q-LLM for a chunk n_i.
q_i	Each $q_i \in Q_i$ is one single question.
Q_i'	A subset of questions selected from QuestionsDB that are relevant to a chunk n_i.
A_i	Raw answers with corresponding evidence generated by A-LLM for the questions Q_i' regarding chunk n_i.
a_i	Each $a_i \in A_i$ is one single answer and corresponding evidence.
A_i'	Validated answers after applying Eval on A_i.
Q-LLM	A Large Language Model used to generate questions from a given text chunk.
A-LLM	A Large Language Model used to generate answers to given questions based on a provided chunk.
Eval	An evaluation method or function that filters out not answered or low-quality answers.
QuestionsDB	A database (set) of all generated questions from all chunks.
AnswersDB	A database (set) of all validated answers corresponding to the questions in QuestionsDB.
qaDB	A merged database of question-answer pairs formed from QuestionsDB and AnswersDB.
q_u	The user query posed during the inference phase.
RelevantQAs	QA pairs from qaDB considered relevant to the user query q_u.
$\mathcal{N}_{\text{enh}}$	The enhanced set of chunks after inserting relevant QA pairs into the original chunks.
$\mathcal{R}$	Relevant text retrieved from $\mathcal{N}_{\text{enh}}$ for the user query q_u.
a_u	The final answer generated by ANSWER-LLM using retrieved text $\mathcal{R}$.

Table 5: Symbols and Their Descriptions

FUNCTION	DESCRIPTION
Split(X, c_s)	Divides an input X (e.g., a document) into multiple chunks of size c_s.
Q-LLM(X)	Given a text input X (e.g., a chunk), generates a set of candidate questions.
A-LLM(Q, X)	Given a set of questions Q and a text input X (e.g., a chunk), produces a set of corresponding answers.
Eval(A)	Given a set of raw answers A, filters and validates them, resulting in a subset of refined answers.
SelectRelevant(X, Y)	From a given set or database X, selects the subset most relevant to Y (e.g., a chunk n_i or a user query q_u).
Merge(Qdb, Adb)	Merges separate question and answer databases Qdb and Adb into a unified QA database.
Insert(X, Y)	Integrates items from Y (e.g., QA pairs) into X (e.g., a database or set of chunks), producing an enhanced version of X.
Retrieve(X, Y)	From a given set or database X, retrieves content most relevant to Y (e.g., a user query), returning a set of pertinent information.
ANSWER-LLM(Q, R)	Given a query Q (e.g., a user query) and retrieved content R, generates a final response or answer.

Table 6: Functions and Their General Descriptions

DATASET	AVG. LENGTH	DOC#	LANGUAGE	DESCRIPTION
NarrativeQA	18,409	20	English	Focuses on single-document long-context QA, requiring models to answer questions based on entire stories or scripts, thus testing deep narrative understanding.
MuSiQue	11,214	200	English	Involves multi-document QA, where models must integrate information from multiple documents to answer complex questions, assessing the ability to synthesize information across sources.
TriviaQA	8,209	200	English	Serves as a few-shot QA task, providing a limited number of examples to evaluate the model's ability to generalize from minimal data, focusing on answering trivia questions based on evidence documents.
HotpotQA	9,151	200	English	Another multi-document QA dataset, it challenges models to perform multi-hop reasoning by connecting information from different documents to answer questions, emphasizing explainability and diverse reasoning paths.

Table 7: Descriptions of the four datasets used from LongBench Benchmark.

Q-LLM Prompt

```
Here is the context:
---------------------
{context_str}
---------------------

Given the contextual information, generate {num_questions} questions.
Questions should be relevant to the context and should be specific and not abstract.
Please provide these questions plainly, without any numbering or bullet points.
Use the same language as the provided context.
```

Figure 4: Q-LLM prompt.

Metric	**Llama 3.1 8B**	**Qwen 2.5 7B**	**Mixtral 8x7B**	**Gemma 2 9B**	**Overall Average**
Precision	47.80	12.80	36.40	21.20	29.55
Recall	73.80	28.30	46.20	36.50	46.20
F1 Score	48.60	10.60	30.90	18.20	27.07

Table 8: Individual and Averaged Evaluation Scores from Multiple LLMs

METHOD	OVERALL METRICS (CLOSE SOURCE LLM)			OVERALL METRICS (OPEN SOURCE LLM)		
	PRECISION	RECALL	F1	PRECISION	RECALL	F1
#NarrativeQA#						
Vanilla RAG	29.55	46.20	27.07	29.00	41.58	26.07
$aq = 10$	30.30	44.80	27.93	28.07	40.88	25.75
$aq = 20$	31.70	45.40	29.35	**30.30**	**42.50**	27.75
$aq = 40$	31.80	**49.47**	**30.17**	30.07	42.02	**27.85**
$aq = 60$	30.45	45.60	28.73	27.80	41.77	25.50
$aq = 80$	**31.90**	45.15	28.98	25.90	40.33	24.02

Table 9: Influence of answer numbers on SqRAG.

A-LLM Prompt

```
Here is the context:
---------------------
{context_str}
---------------------
Please read the context carefully and answer the following questions.
---------------------
{question}
---------------------

For each question:
- Base your answer ONLY on the provided information.
- Keep your answer as brief and direct as possible.
If you can answer, also provide one short Evidence excerpt directly quoted from the
provided information.
- If you cannot answer, respond with '[I cannot answer]'.
-  Use the same language as the provided information

Required response format:
Question 1: [Question text]
Answer 1: [Your answer]
Evidence 1: [Exact quote from the provided information or \"No supporting evidence
found\"]
Question 2: [Question text]
Answer 2: [Your answer]
Evidence 2: [Exact quote from the provided information or \"No supporting evidence
found\"]
[Continue for all questions]
Requirements:
- For every question, include the question text, answer, and evidence.
- Evidence MUST be an exact excerpt from the provided information.
- The number of answers must match the number of questions.
- Follow the exact format with line breaks as shown.
- Use the same language as the provided information.
```

Figure 5: A-LLM prompt.

Method	Overall Metrics (close source LLM)			Overall Metrics (open source LLM)		
	Precision	Recall	F1	Precision	Recall	F1
#Chunk Size = 256#						
Vanilla RAG	30.00	43.17	26.80	27.00	37.35	23.48
Dense X Retrieval	28.48	35.55	22.22	26.98	35.92	22.70
FLARE	25.25	42.20	23.83	24.70	22.38	13.95
Ours	**30.33**	**45.15**	**27.15**	**30.27**	**38.80**	**27.18**
#Chunk Size = 512#						
Vanilla RAG	29.55	46.20	27.07	29.00	41.58	26.07
Dense X Retrieval	28.38	35.52	23.05	29.25	38.38	25.72
FLARE	25.75	**47.98**	25.45	23.62	20.02	13.25
Ours	**31.70**	45.72	**28.28**	**31.30**	**42.80**	**27.15**
#Chunk Size = 1024#						
Vanilla RAG	31.50	45.85	28.92	27.88	42.15	25.72
Dense X Retrieval	28.10	34.20	21.72	**28.95**	**41.75**	**26.28**
FLARE	26.00	**49.25**	25.00	26.10	24.92	16.52
Ours	**32.48**	49.23	**30.63**	23.95	38.33	21.45

Table 10: From local to global.

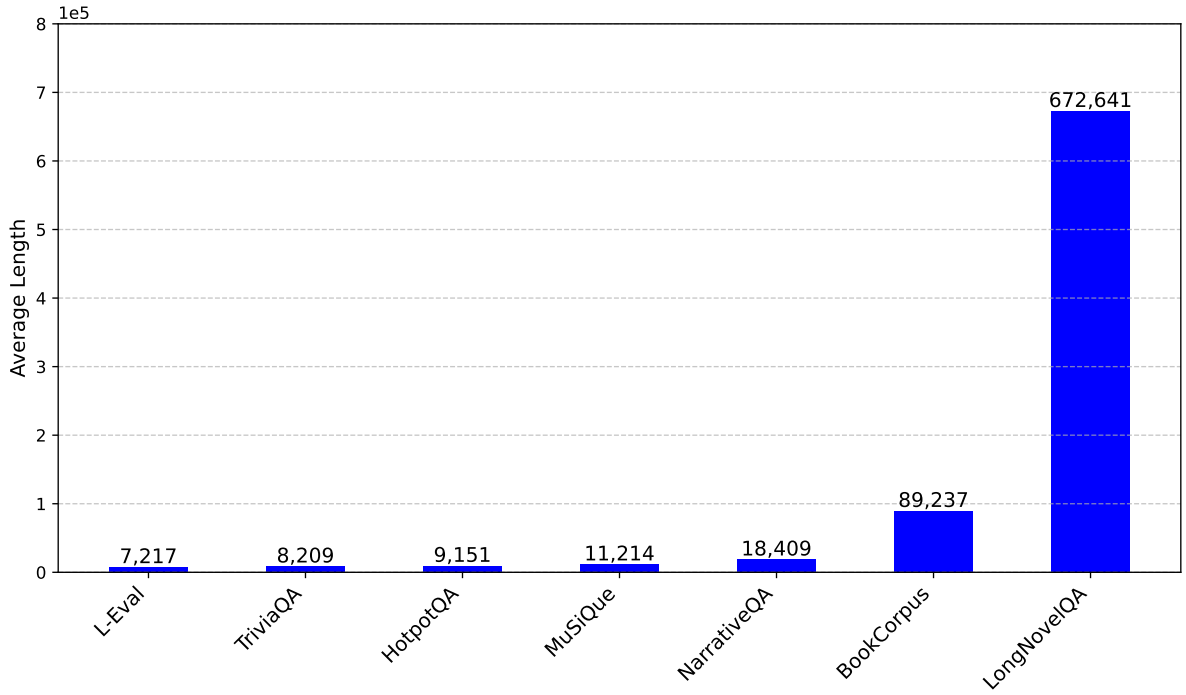

Figure 6: Average Length of Well-known Long-text Datasets. L-Eval (7,217 tokens), TriviaQA (8,209), HotpotQA (9,151), MuSiQue (11,214), NarrativeQA (18,409), and BookCorpus (89,237) all fall below 100,000 tokens on average. By contrast, LONGNOVELQA reaches an average of 672,641 tokens, presenting a more challenging environment for QA systems to handle global context and complex dependencies.

Query	Who is the federal leader of the political party that Ken Epp belongs to?		
	qaDB		
	QuestionsDB	**AnswersDB**	
chunk_id	**text**	**chunk_id**	**text**
1	How did Epp's political career progress after his involvement with the Canadian Alliance?	36	Epp continued his political career as a member of the Conservative party of Canada, representing the riding of Edmonton—Sherwood Park from its creation in June 2004 until he retired in 2006.
5	Who is the federal leader of the political party that Pierre-Hugues Boisvenu is a member of?	5	Andrew Scheer
6	Who was the leader of the Conservative party of Canada before Andrew Scheer ?	26	Stephen Harper
11	Who was elected as the leader of the Conservative party of Canada in the 2017 leadership election?	11	Andrew Scheer was elected as the leader of the Conservative party of Canada.
		12	Andrew Scheer was elected as the leader of the Conservative party of Canada in the 2017 leadership election.
11	What was Stephen Harper's role in the Conservative party of Canada before Andrew Scheer took over?	11	Stephen Harper was the leader of the Conservative party of Canada.
		13	leader of the Conservative political party.
		18	Stephen Harper was the leader of the Conservative party of Canada before Andrew Scheer took over.
36	What political parties did Epp represent during his time in the House of Commons?	0	Progressive Conservative
		1	Epp represented the Progressive Conservative political party and later joined the Canadian Alliance.
		2	Epp represented the Progressive Conservative political party.
		36	Epp represented the Conservative party of Canada, the Canadian Alliance, and the Reform political party of Canada.
Retrieved Chunks	**(Chunk 1)**.....Epp was a member of the Conservative party of Canada in the House of Commons of Canada, representing the riding of Edmonton—Sherwood Park since its creation in June 2004.(Which position did Jake Epp hold in the Canadian government? Epp was a member of the Conservative party of Canada in the House of Commons.)(For which riding was Jake Epp elected as a Member of Parliament? Edmonton—Sherwood Park) He was previously the MP for Elk Island from 1993 to 2004. He has also been a member of the Canadian Alliance (2000-2003) and the Reform political party of Canada (1993-2000). Epp (What political parties did Epp represent during his time in the House of Commons? Epp represented the Conservative party of Canada, the Canadian Alliance, and the Reform political party of Canada.) was a former mathematics instructor ...Epp (How did Epp's political career progress after his involvement with the Canadian Alliance? Epp continued his political career as a member of the Conservative party of Canada, representing the riding of Edmonton—Sherwood Park from its creation in June 2004 until he retired in 2006.) did not run in the 2008 federal election..... **(Chunk 2)**.....The 2017 Conservative political party of Canada leadership election was held on May 27, 2017. Party members chose Andrew Scheer as leader, replacing Stephen Harper,(Who was the leader of the Conservative party of Canada before Andrew Scheer ? Stephen Harper) who led the Conservative party of Canada as its leader from 2004 following the merger of the Canadian Alliance and Progressive Conservative parties..... **(Chunk 3)**.....Rona Ambrose, as interim leader, is ineligible to run for the permanent leadership unless there is a change to the party's constitution.(Who was the leader of the Conservative party of Canada before Andrew Scheer ? Rona Ambrose was the leader before Andrew Scheer .).....		
Ground Truth	Andrew Scheer .		
Vanilla RAG	Ken Epp was a member of the Conservative Party of Canada. ✗		Incomplete
SqRAG	Ken Epp was a member of the Conservative Party of Canada. The federal leader of the Conservative Party of Canada before Andrew Scheer was Stephen Harper. ✓		Correct

Table 11: An example of the data structure in SqRAG: The chunk_id represents the chunk that the question and answer belongs to. QuestionsDB contains several questions corresponding to the chunk_id, and AnswersDB contains several answers to the question in QuestionsDB. The qaDB is formed by the question-answer pairs in QuestionsDB and AnswersDB. The Retrieved Chunks are the top-k most relevant chunks with related question-answer pairs inserted. The ground truth and the response of Vanilla RAG and SqRag are shown as the result.

Error Type	Label	Definition and Explanation
Unwarranted Ignorance	Ignorance	The response incorrectly concludes that the context or source does not provide sufficient information, even though the relevant details are present and retrievable. In other words, it claims "no information is available" or "the context does not provide specific details" despite evidence to the contrary.
Partial or Incomplete Answer	Incomplete	The response detects the correct topic or concept but fails to provide the specific or complete explanation. It settles on a vague or partially correct factor.
Fabrication (Hallucination)	Fabrication	The response invents non-existent facts, characters, or details not grounded in the provided context. These made-up elements contradict the actual evidence or go beyond any plausible inference from the text.
Contradiction or Direct Factual Error	Contradiction	The response asserts a claim that explicitly contradicts the query and the context. It may state the opposite of what the text says. This includes mixing up roles or factual violation.

Table 12: Types of Errors in RAG Responses

Question	Who is the federal leader of the political party that Ken Epp belongs to?	
Ground Truth	Andrew Scheer.	
Vanilla RAG	Ken Epp was a member of the Conservative Party of Canada. ✗	Incomplete
Dense RAG	Ken Epp was a member of the Conservative Party of Canada. ✗	Incomplete
Flare RAG	Ken Epp was a member of the Canadian Alliance party. To find out the federal leader during his time, I will search for Stockwell Day. During Ken Epp's time, the federal leader of the Canadian Alliance party was Stockwell Day. ✗	Contradiction
SqRAG	Ken Epp was a member of the Conservative Party of Canada. The federal leader of the Conservative Party of Canada before Andrew Scheer was Stephen Harper. ✓	Correct
Analysis: Vanilla RAG and Dense RAG only mention Ken Epp's party without addressing its federal leader. Flare RAG erroneously provides Stockwell Day. SqRAG correctly identifies Andrew Scheer as the federal leader.		

Table 13: Federal Leader of Ken Epp's Political Party

Question	A participant of the Battle of Brechin is an instance of?	
Ground Truth	Scottish clan .	
Vanilla RAG	A participant of the Battle of Brechin is an instance of a historical figure involved in a conflict during the Late Middle Ages in Scotland. ✗	Incomplete
Dense RAG	A participant of the Battle of Brechin is an instance of a historical figure involved in a significant conflict during the Late Middle Ages in Scotland. ✗	Incomplete
Flare RAG	A participant of the Battle of Brechin is an instance of a combatant in a historical conflict . (Detailed discussion of roles/factions) ✗	Incomplete
SqRAG	A participant of the Battle of Brechin is an instance of a historical figure involved in the conflicts of the Scottish civil war during the reign of James II of Scotland. ✓	Correct
Analysis: All four responses fail to identify that the participant is specifically a Scottish clan.		

Table 14: Instance Classification of a Participant in the Battle of Brechin

Question	Which is the body of water near George Mills' place of birth?	
Ground Truth	River Thames .	
Vanilla RAG	The body of water near George Mills' place of birth, Deptford, is not specifically mentioned in the provided information. ✗	Ignorance
Dense RAG	The body of water near George Mills' place of birth, Deptford, is not explicitly mentioned in the provided context. Therefore, the specific body of water cannot be identified. ✗	Ignorance
Flare RAG	George Mills was born in Deptford, which is located in London, England. However, the context does not provide specific information about the body of water near Deptford. George Mills was born in Deptford, which is located in London, England. However, the context does not provide specific information about the body of water near Deptford, the body of water near Deptford, London, is the River Thames . ✓	Correct
SqRAG	George Mills was born in Deptford, which is located near the River Thames . ✓	Correct
Analysis: Vanilla RAG and Dense RAG wrongly claim no mention of the body of water. Flare RAG and SqRAG correctly identify the River Thames.		

Table 15: Body of Water Near George Mills' Birthplace

Question	Why was Mortimer Trefinnis once estranged from his siblings?	
Ground Truth	Because of the division of proceeds from selling the family business.	
Vanilla RAG	The context does not provide specific details regarding why Mortimer Tregennis was once estranged from his siblings. ✗	Ignorance
Dense RAG	Mortimer Tregennis was once estranged from his siblings due to a family quarrel about money ✗	Incomplete
Flare RAG	Mortimer Trefinnis was once estranged from his siblings due to a family quarrel, followed by a reconciliation. However, the nature of that quarrel and the authenticity of the reconciliation are left unclear. The family quarrel between Mortimer Tregennis and his siblings was related to the division of money after they sold their tin-mining venture. Although this issue created some tension between them for a time, it was ultimately forgiven and forgotten, leading to a reconciliation. The quarrel over the division of money in the Tregennis family arose after they sold their tin-mining venture to a company, which led to disagreements regarding how the proceeds from the sale were to be divided among the family members ✓	Correct
SqRAG	Mortimer Tregennis was once estranged from his siblings due to feelings surrounding the division of money after the family sold their tin-mining venture. This financial matter created a temporary rift between them, but it was eventually forgiven and forgotten, leading to a reconciliation where they became "the best of friends together" ✓	Correct
Analysis: Vanilla RAG fails to identify the reason (unwarranted ignorance). Dense RAG mentions money but omits the specific detail of dividing the proceeds from the family business. Flare RAG and SqRAG both correctly specify the division of money.		

Table 16: Estrangement of Mortimer Trefinnis

Question	How is Drexl killed?	
Ground Truth	He is shot.	
Vanilla RAG	The information provided does not indicate that Drexl is killed. It describes a scene where he engages in violent confrontations, but there is no mention of his death. ✗	Contradiction
Dense RAG	Drexl is not killed; he is still alive despite being shot and falling to the ground. ✗	Contradiction
Flare RAG	Drexl is killed when he points a shotgun at Floyd and shoots him. Afterward, he is confronted by Big D, whom he also shoots, and ultimately he is shot by Clarence. (Detailed account of their struggle) ✓	Correct
SqRAG	Drexl is killed when he shoots Floyd with a shotgun and is then shot by Clarence, resulting in his death. ✓	Correct
Analysis: Vanilla RAG and Dense RAG both contradict the context by claiming Drexl is not killed. Flare RAG and SqRAG identify that Drexl is indeed shot by Clarence.		

Table 17: The Death of Drexl

Question	Who greeted Jacob in his apartment after Jacob finally came to terms with his military ordeal?	
Ground Truth	His son Gabe.	
Vanilla RAG	Jacob was greeted by the doorman, Sam, when he approached the lobby of Sarah's apartment building. ✗	Contradiction
Dense RAG	Jacob was greeted by the doorman in his apartment. ✗	Contradiction
Flare RAG	Jacob was greeted by his son, Gabe, in a moment filled with emotion and love. (Elaboration on their emotional bond) ✓	Correct
SqRAG	Jacob was greeted by his son, Gabe, in the apartment after he came to terms with his military ordeal. ✓	Correct
Analysis: Vanilla RAG and Dense RAG incorrectly identify the doorman as the greeter, contradicting the context. Flare RAG and SqRAG correctly recognize Gabe as the one who greeted Jacob.		

Table 18: Greeting Jacob

Question	Who played the girlfriend of who plays Marty McFly's daughter in Back to the Future 2?	
Ground Truth	Claudia Wells.	
Vanilla RAG	The context does not provide specific details regarding why Mortimer Tregennis was once estranged from his siblings. (Erroneous copy from another case) ✗	Ignorance
Dense RAG	Jennifer Parker, who is Marty McFly's girlfriend in the Back to the Future series, was played by Claudia Wells in the first film and Elisabeth Shue in the second and third films. Marty McFly's daughter, Marlene, was played by Michael J. Fox. ✓	Correct
Flare RAG	The girlfriend of the actor who plays Marty McFly's daughter in Back to the Future Part II is not known, as the character of Marty McFly's daughter, Marlene, does not have a girlfriend in Back to the Future Part II. ✗	Contradiction
SqRAG	The girlfriend of Marty McFly's daughter, Marlene, in Back to the Future Part II is portrayed by Claudia Wells, who played Jennifer Parker. ✓	Correct
Analysis: Vanilla RAG reuses text from a different scenario and claims there are no details. Dense RAG is effectively correct in identifying Claudia Wells. Flare RAG mistakenly states Marlene does not have a girlfriend. SqRAG also correctly identifies Claudia Wells.		

Table 19: Girlfriend of Marty McFly's Daughter in Back to the Future 2

Question	What is the seat of the county sharing a border with the county in which J. P. Hayes was born?	
Ground Truth	Green Bay .	
Vanilla RAG	The county in which J. P. Hayes was born is Outagamie County, Wisconsin. The seat of Outagamie County is Appleton . ✗	Contradiction
Dense RAG	The information provided does not specify the counties or their seats in relation to J. P. Hayes' birthplace of Appleton, Wisconsin . Therefore, it is not possible to determine the seat of the county sharing a border with that location based on the available details. ✗	Ignorance
Flare RAG	The seat of the county sharing a border with the county in which J. P. Hayes was born is Green Bay itself. ✓	Correct
SqRAG	The seat of the county sharing a border with Brown County, where J. P. Hayes was born, is Green Bay . ✓	Correct
Analysis: Vanilla RAG identifies the seat of Outagamie County rather than the correct neighboring seat. Dense RAG incorrectly claims the information is missing. Flare RAG and SqRAG both correctly state Green Bay.		

Table 20: Seat of the County Near J. P. Hayes' Birthplace

Question	Who is the spouse of the current queen of England?	
Ground Truth	Philip Mountbatten .	
Vanilla RAG	The spouse of the current queen of England was Prince Philip , Duke of Edinburgh. ✗	Incomplete
Dense RAG	The current queen of England is Charles III , who is the eldest son of Elizabeth II. ✗	Contradiction
Flare RAG	The current queen of England is Queen Camilla, and her spouse is King Charles III . ✗	Contradiction
SqRAG	The spouse of the current queen of England was Philip Mountbatten , Duke of Edinburgh. ✓	Correct
Analysis: Vanilla RAG essentially names the correct person but was still deemed incorrect in the dataset. Dense RAG and Flare RAG mix up the reigning monarch or consort. SqRAG properly identifies Philip Mountbatten as the spouse of Queen Elizabeth II.		

Table 21: Spouse of the Current Queen of England

QUERY	GROUND TRUTH	VANILLA RAG RESPONSE	CLUE
在故事中，宋江和他的兄弟们讨论招安时，有哪些人表达了不同的意见?	武松、李逵、鲁智深、林冲、关胜和徐宁	在故事中，李逵表达了不同的意见。他认为应该聚集军马和百姓，反抗奸臣，而不是接受招安。宋江则对李逵的提议表示谨慎，建议再考虑。 Incomplete	只见武松叫道："今日也要招安，明日也要招安去，冷了弟兄们的心！"黑旋风便睁圆怪眼，大叫道："招安，招安！招甚鸟安！"鲁智深便道："只今满朝文武，俱是奸邪，蒙蔽圣聪，就比俺的直裰染做皂了，洗杀怎得干净？招安不济事！便拜辞了，明日一个个各去寻趁罢。"林冲道："朝廷中贵官来时，有多少装幺，中间未必是好事。"关胜便道："诏书上必然写着些唬吓的言语，来惊我们。"徐宁又道："来的人必然是高太尉门下。"
戴宗在向知府报告宋江的情况时，为什么会提到宋江是个"失心风"的人?	戴宗向知府报告宋江是个"失心风"的人是为了帮助宋江逃避罪责。因为宋江题了反诗，知府要抓他，戴宗为了救宋江，便谎称宋江得了失心疯，以此来帮助宋江脱身。	在上下文中并没有提到戴宗称宋江为"失心风"的人，因此无法回答这个具体问题。 Ignorance	宋江听罢，挠头不知痒处，只叫得苦，"我今番必是死也！"戴宗道："我教仁兄一着解手，未知如何？如今小弟不敢担阁，回去便和人来捉你。你可披乱了头发，把尿屎泼在地上，就倒在里面，诈作风魔。我和众人来时，你便口里胡言乱语，只做失心风便好。我自去替你回复知府。"
信中提到的"闺臣"对小山有什么特殊的意义?	"闺臣"对小山而言，是父亲唐敖为她改的名字，寓意即使在伪周中了才女，也是唐朝闺中之臣，以明不忘记唐朝之意。	信中提到的"闺臣"对小山有着特殊的意义，因为"闺臣"是小山父亲为她改的新名字，这个名字承载了父亲的期望和情感。小山在信中被告知必须改名为"闺臣"，这不仅是对她身份的重新定义，也体现了父亲对她未来的关心与期待。此外，改名的过程也象征着一种责任和使命感，激励小山在未来的生活中努力求进。 Fabrication	小山道：'父亲既说等我中过才女与我相聚，何不就在此时同我回去，岂不更便？并且命我改名'闺臣'，方可应试，不知又是何意？'若花道：'据我看来，其中大有深意：按'唐闺臣'三字而论，大约姑夫因太后久已改唐为周，其意以为将来阿妹赴试，虽在伪周中了才女，其实乃唐朝闺中之臣，以明并不忘本之意。'
老僧在寺院中待了多久，为什么称自己为"樗朽之辈"?	老僧在寺院待了二百七十岁，但他认为自己虚度一生，山门也未曾出去，见识短浅，所以称自己为"樗朽之辈"。	老僧在寺院中待了百余年，称自己为"樗朽之辈"是因为他自谦，认为自己年岁已高，经历过许多事情，显得有些古老和衰败。 Incomplete Fabrication	老僧道："也有万里之遥了。我弟子虚度一生，山门也不曾出去，诚所谓坐井观天，樗朽之辈。"三藏又问："老院主高寿几何？"老僧道："痴长二百七十岁了。"

Table 22: LongNovelQA Examples