Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.082460

ARTICLE

Chronological Passage Assembly for Retrieval-Augmented Generation in Narrative Question Answering

Byeongjeong Kim, Jeonghyun Park, Joonho Yang and Hwanhee Lee*

Department of Artificial Intelligence, Chung-Ang University, Seoul, Republic of Korea

*Corresponding Author: Hwanhee Lee. Email: hwanheelee@cau.ac.kr

Received: 16 March 2026; Accepted: 04 June 2026; Published: 23 July 2026

ABSTRACT: Long-context question answering over narrative documents remains challenging because many questions require reconstructing event sequences while preserving local contextual flow under limited context budgets. Existing retrieval-augmented generation (RAG) methods typically retrieve document snippets independently, which can fragment narratives and harm temporal dependencies. We propose ChronoRAG, a retrieval framework for narrative question answering that first converts sequential document chunks into concise relation descriptions and then retrieves relevant units together with their adjacent chronological context. This design preserves retrieval precision while providing the generator with coherent local narrative structure. Experiments on NarrativeQA and GutenQA show that ChronoRAG improves performance on NarrativeQA and remains competitive on GutenQA, with particularly strong gains on questions that require chronology-sensitive context. These results suggest that explicitly modeling local event order is a useful retrieval signal for narrative question answering.

KEYWORDS: Retrieval-augmented generation; narrative question answering; long-context reasoning; temporal reasoning; knowledge graphs

1 Introduction

Long-context question answering tasks, which require the ability to utilize one or more long documents [1], present a significant challenge in natural language processing. While modern transformer-based Large Language Models (LLMs) have shown a remarkable ability to handle long contexts [2,3], they face fundamental limitations when confronted with extremely long-form text. Processing extensive documents for every query leads to major computational inefficiency, and as the context grows longer, the models' ability to accurately identify and prioritize relevant information decreases, impacting the reliability of their outputs.

To address these challenges, Retrieval-Augmented Generation (RAG) [4] has become a standard approach, focusing on efficiently retrieving only relevant segments from large documents to integrate into the model's context window. This selective retrieval method helps models leverage vast knowledge bases far beyond their built-in context limits.

However, a fundamental methodological gap exists in most RAG frameworks [4,5]: they primarily treat documents as a collection of short, independently-retrieved snippets of information. This methodology fundamentally conflicts with the sequential nature of long-form narratives, such as those found in history, literature, and film. Narrative texts are uniquely defined by their structure; they can be extremely long, their individual passages often fail to convey the full story unless read in order, and grasping the chronological and

relational connections between passages is essential for comprehension. Treating passages as isolated facts severs these critical links, fragmenting the narrative timeline.

Fig. 1 illustrates the mismatch between conventional retrieval strategies and the characteristics of narrative text. As shown in Fig. 1a, a common approach is to retrieve as many sentences as possible that are likely to match the query based on textual similarity. To do so, documents are typically stored as isolated sentences. While such methods may successfully retrieve a sentence containing the correct answer, they often fail to provide sufficient contextual cues. This can create ambiguity, making it unclear whether "London" or "Paris" is the location relevant to the question, even if both are mentioned in the retrieved results. In narrative question answering, the meaning of an event is rarely determined by a single sentence alone; rather, it depends on how that event is situated within the surrounding sequence of events. Consequently, even when retrieval succeeds at locating a relevant fact, it may still fail to recover the local narrative context required to interpret that fact correctly.

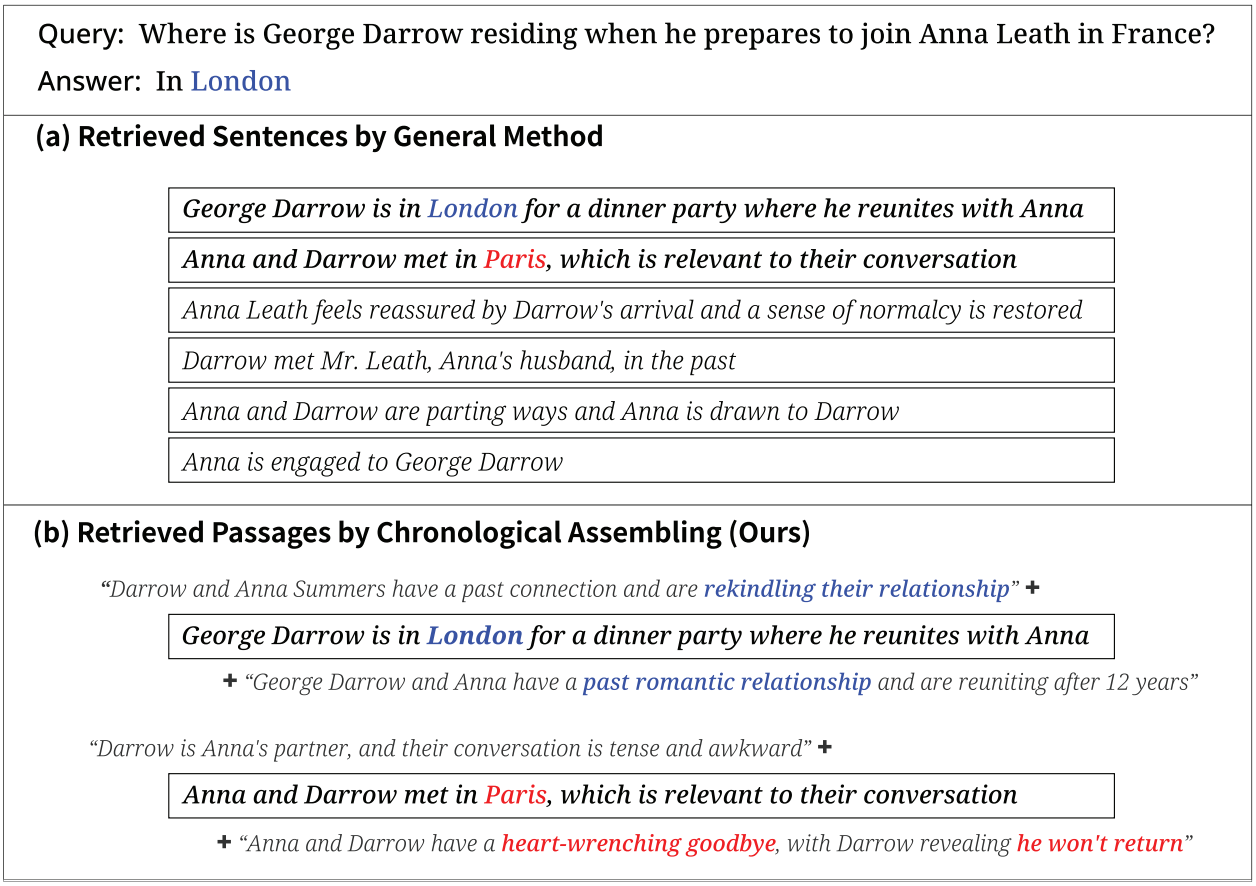

Figure 1: Retrieval comparison for a narrative query. (**a**) Fine-grained indexing returns six standalone sentences, leaving key clues detached. (**b**) Our chronological assembling retrieves passages that include their immediate chronological context, preserving the narrative flow. Boxes indicate the directly retrieved sentences.

To address this issue, we introduce ChronoRAG, a novel RAG-based approach grounded in the observation that narrative question answering fundamentally requires preserving the chronological organization of events. ChronoRAG addresses this challenge by separating retrieval precision from contextual reconstruction: it first retrieves fine-grained relation units as precise retrieval keys, and then reassembles their adjacent chronological context to restore the local meaning of the retrieved evidence. Instead of maximizing

the number of retrieved sentences, our framework, as shown in Fig. 1b, retrieves fewer distinct informational units but includes their surrounding context to disambiguate meaning. This approach provides the crucial contextual clues—indicating that "London" is associated with a reunion while "Paris" pertains to a farewell—that are essential for accurate question answering. ChronoRAG achieves this by transforming dispersed narrative content into structured passages while preserving cues that help recover the local narrative flow, enabling the retrieval of a coherent sequence of evidence rather than a collection of isolated facts.

We empirically validate our proposed approach on the NarrativeQA [6] and GutenQA [7]. For auxiliary analysis, we isolate a subset of "Time Questions" subset containing questions with explicit temporal markers. Our experiments show that our method achieves significant improvements in both the complete dataset and the specialized Time Question set. Notably, these results are achieved using lighter graph construction and retrieval mechanisms than those found in existing summary and graph-based methods, demonstrating enhanced performance in identifying individual facts and comprehending complex relational structures.

Our contributions are as follows:

- We introduce a local chronology-aware retrieval unit, in which a fine-grained relation description serves as an atomic retrieval key and its immediate narrative neighbors are dynamically assembled as supporting context, preserving local coherence without sacrificing retrieval precision.
- We introduce a novel RAG framework, ChronoRAG, which refines raw text into structured passages, explicitly maintains temporal links between events, and incorporates adjacent context.
- Through benchmark experiments and ablation studies, we show that local chronological linking improves answer quality, especially on questions that benefit from chronology-sensitive context.

2 Related Work

2.1 Long Context Question Answering

Long-context question answering (LCQA) refers to the task of answering questions over documents that are too long to be directly processed in full under the target inference setting [1,8,9]. Unlike conventional question answering, which assumes that the full input can be handled at once, LCQA addresses settings in which the source document exceeds the practical context budget, requiring the system to selectively access, compress, or structure evidence before answer generation. This setting commonly arises in domains involving long-form text, such as books, legal documents, and narratives, where questions may range from localized fact lookup to broader contextual interpretation over surrounding discourse. Prior work on LCQA can be broadly categorized into two directions: (1) approaches that expand the amount of context that can be processed directly by the model, and (2) retrieval-based methods that select and process only a subset of the available context. These two paradigms reflect different strategies for addressing long-context question answering under finite context and computational budgets.

2.2 Long-Context Modeling Approaches

One line of research addresses LCQA by expanding the amount of context that can be processed directly by the model. Prior work has proposed architectures with more efficient attention mechanisms, such as Longformer [10] and BigBird [11], and more recent long-context language models have further increased the amount of text that can be handled in a single input. These approaches reduce the need for external retrieval by allowing the model to access longer spans of source text directly.

However, when source documents are very long, repeatedly processing the full document for each question can still be computationally expensive and may introduce substantial distractor content, especially when only a small portion of the document is relevant. These trade-offs motivate alternative approaches to long-context question answering, which we discuss next.

However, when source documents are very long, repeatedly processing the full document for each question can still be computationally expensive and may introduce substantial distractor content, especially when only a small portion of the document is relevant. These trade-offs have motivated retrieval-based approaches, which aim to identify and process only the evidence most useful for answering a given question.

2.3 RAG Based Long Context Question Answering

An alternative line of work addresses long-context question answering through retrieval rather than direct full-context processing. Instead of increasing the amount of text consumed by the model, retrieval-based methods identify and process only the evidence most useful for answering a given question. This paradigm is valuable not only for efficiency, but also for evidence selection and noise reduction when only a limited portion of a long document is relevant. In this setting, many methods improve retrieval by transforming source documents into structured intermediate representations, often generated or refined with the help of LLMs, that make long-context evidence easier to retrieve and organize.

Existing RAG-based approaches for long-context QA have developed along several directions. One line of work studies retrieval granularity, moving from document- or passage-level retrieval toward finer-grained units such as propositions or relations. Another line uses summarization or abstraction to compress source content into more retrieval-efficient representations. A third line incorporates graph-structured or knowledge-aware representations to better model dependencies among retrieved units. ChronoRAG is most closely related to this retrieval-side line of work: rather than increasing the model context window, it seeks to improve how long narrative documents are structured and retrieved under a fixed evidence budget.

2.3.1 Retrieval Granularity

Document indexing approaches have explored a wide range of retrieval granularities. DenseXRetrieval [12] shows that proposition-level indexing can improve retrieval precision by transforming text into fine-grained, self-contained factoid units. At the same time, MolecularFacts [13] argues that fully atomic decompositions often lose the context needed to interpret a claim correctly, and therefore advocates minimally decontextualized units that remain understandable in isolation. Our work shares this concern with retrieval unit design, but differs in how it conceptualizes fine-grained evidence. We do not treat fine-grained relations as an unordered collection of independent units. Instead, we view them as a sequence of locally dependent narrative states. Accordingly, ChronoRAG indexes relation descriptions at fine granularity for precise matching, but retrieves them together with their adjacent neighbors so that the local order of the story is preserved. In narrative QA, this distinction matters because the answer often depends not only on which fact is retrieved, but also on what immediately precedes and follows it.

2.3.2 Summary-Based Document Augmentation

A second line of work augments documents through hierarchical compression. RAPTOR [5] recursively clusters and summarizes chunks into a tree, enabling retrieval across multiple levels of abstraction. MemWalker [14] and A Human-Inspired Reading Agent [15] likewise use summary structures or gist memories to navigate long contexts efficiently. These approaches demonstrate that abstraction can substantially improve long-document retrieval by filtering irrelevant material and exposing higher-level document structure. However, this strength can also be a limitation for narrative QA, where answers often depend on fine-grained local details. Repeated summarization is designed to preserve salient themes, not necessarily the specific details that align with the wording of a question. In addition, recursive summarization may amplify recurring information across levels, creating redundancy while overlooking sparse but answer-critical details. As a result, important narrative cues may be lost even when the global summary remains

faithful to the overall story. ChronoRAG therefore adopts only a shallow summarization step and preserves access to the original source chunks so that such details remain recoverable at inference time.

2.3.3 Knowledge Graph-Based Document Augmentation

Graph-based augmentation introduces explicit relational structure into retrieval. Conceptually, many methods in this family resemble an Open Information Extraction [16]-style decomposition of text into atomic relational statements, where entities function as nodes and relations function as edges. GraphRAG [17] builds an entity knowledge graph and community summaries to support graph-based retrieval and question-focused aggregation, while LightRAG [18] improves efficiency through graph-structured dual-level retrieval. These approaches are effective when the key challenge is exposing latent relations beyond flat chunk similarity. However, they mainly reorganize text into entity-centered neighborhoods rather than preserving the original order in which the narrative unfolds.

Recent work has begun to address this limitation more directly. EventRAG [19] decomposes text into interconnected events and organizes them in an event knowledge graph with temporal-aware reasoning. Entity–Event RAG [20] separates entity and event subgraphs and links them through a bipartite mapping so that temporal and causal information is not collapsed into a single entity node. KG-IRAG [21] further extends graph-based retrieval through iterative reasoning over temporal and logical dependencies. These methods are important steps toward chronology-aware retrieval, but they still start by transforming the story into graph components and then reasoning over those components. ChronoRAG takes a different view. Rather than first dissolving the narrative into graph elements and later reconstructing chronology, we preserve the linear discourse order itself as part of the index by linking adjacent fine-grained relation units according to their original narrative sequence. In this sense, our graph is not only relational but also sequence-preserving, which makes it particularly suited to narrative question answering.

A related line of work in graph-based question answering uses the graph itself as the substrate for answering queries. Within this paradigm, some methods answer questions by directly retrieving a relevant local portion of the graph, whereas others decompose the query into relational chains and solve it through multi-hop reasoning [22–24]. ChronoRAG follows the former, single-hop setting. Our focus is not on constructing long reasoning chains over a knowledge graph, but on preserving narrative flow within a single retrieval step.

3 Proposed Method

We present ChronoRAG, a novel RAG framework specialized for narrative texts where chronological context is crucial. Most RAG systems treat documents as a collection of independent facts, which fragments the timeline and severs the contextual links essential for understanding sequential events. To address this, we design our framework to reconstruct narrative flow by explicitly modeling and preserving the temporal order of events. As described in Fig. 2, our framework is composed of two primary stages: an offline Graph Construction phase where the original documents are processed into a hierarchical, linked structure, and an online Passage Retrieval and Answer Generation phase where the constructed graph is used to answer queries.

3.1 Offline Graph Construction

This offline phase transforms a raw document into a structured, two-layer graph that captures both factual information and narrative chronology. As shown in Fig. 2, this graph is composed of two levels: a foundational ***Layer 0*** containing the original document text divided into sequential, fixed-length chunks,

which preserves narrative detail, and an abstract ***Layer 1*** built from concise, structured relation descriptions that represent the key events and relationships. The graph construction process involves four steps.

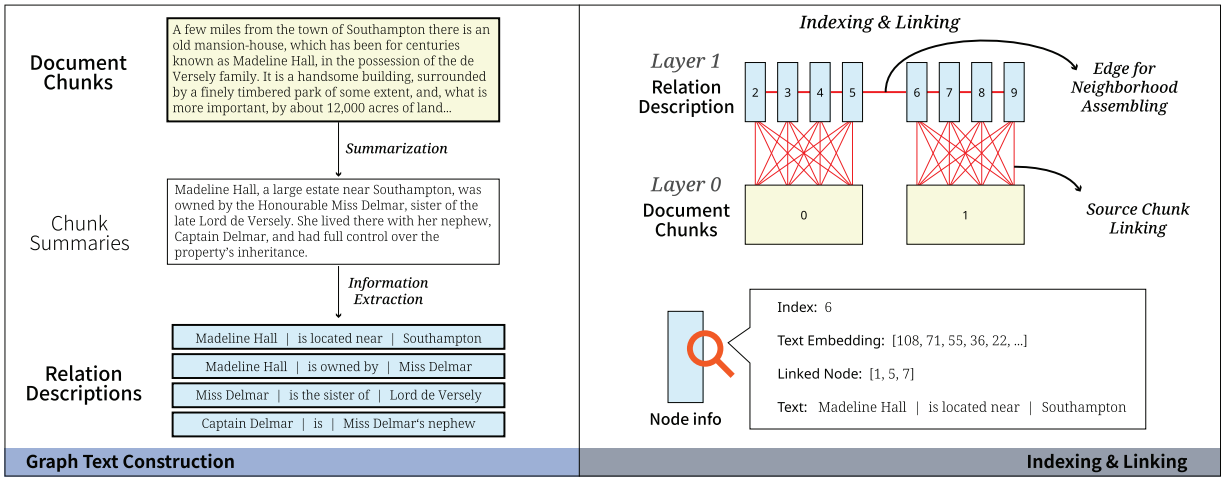

Figure 2: The offline graph construction pipeline of ChronoRAG. This process transforms an unstructured narrative document into a structured, two-layer graph that explicitly encodes chronological relationships.

3.1.1 Document Chunking

Due to inherent limitations in processing an entire document simultaneously, we first divide the original document (D) into fixed-length chunks (d_i), each consisting of up to k tokens. This approach ensures that all retrieved document segments fit within a predefined context length, thereby maintaining both the quality and manageability of retrieval results. The document is initially segmented into individual sentences, which are then sequentially appended to each chunk. When the cumulative length exceeds k tokens, the next sentence is assigned to a new chunk, ensuring that sentences are not split across chunks. In rare cases where a single sentence itself exceeds k tokens, the sentence is split to guarantee that every chunk remains within the token limit. These segmented document chunks serve as the fundamental retrieval units and constitute *Layer 0* of the graph constructed in subsequent stages. The document D is thus represented as a set of chunks, where each chunk d_i has a token count less than k, as follows:

$$D = \{d_1, d_2, d_3, \ldots, d_i\}, \quad |d_i| < k \tag{1}$$

3.1.2 Chunk Summarization

Next, we sequentially cluster the chunks in the original document order, grouping every l chunks. We concatenate and summarize each cluster's texts using an LLM. We utilize sequential clustering because it preserves the document's original order while maintaining a controllable and consistent input length for the LLM. This summarization step facilitates higher-level representation learning by focusing on the overall flow of the document rather than retaining excessive local detail. For example, the left panel of Fig. 2 shows how a raw *Document Chunk* about *Madeline Hall* is condensed into a more concise *Chunk Summary*. For each cluster of l chunks, we generate a summary S_i by an LLM using a summarization prompt $P_{\text{summarize}}$ as formulated below (We provide a full prompt in Appendix A):

$$s_i = \text{LLM}(P_{\text{summarize}}, \{d_{i1}, d_{i2}, d_{i3}, \ldots, d_{il}\}) \tag{2}$$

3.1.3 Entity-Relation Extraction

We then transform the generated summaries via LLM into relational descriptions among entities. We adapt the prompt of GraphRAG [17] into a one-shot instruction for entity–relation extraction (Full prompts are in Table A1 of Appendix A). From LLM's outputs, which consist of both entity descriptions (E_i) and relation descriptions(R_i), we only use the relation descriptions to form the *Layer 1* nodes of the graph. This extraction step decomposes the summarized text into retrieval-friendly fragments, as described in the left panel of Fig. 2, where the summary is broken down into atomic facts like "*Madeline Hall is owned by Miss Delmar*".

Here, we exclude entity descriptions because identical entities may appear redundantly across multiple chunks. Finally, we formalize this extraction step, where an LLM processes each summary S_i to produce a set of entities E_i and relations R_i, as shown below (Full prompts are in Appendix A):

$$\{E_i, R_i\} = \mathrm{LLM}(P_{\mathrm{extraction}}, S_i) \tag{3}$$

3.1.4 Hierarchical and Temporal Indexing

This step involves assigning indices to the relation description sentences (R_i) derived from the summary and the document chunks (d_i) from the original text. Document chunks are indexed sequentially according to their original order in the source text, while the relation description sentences are indexed either based on the earlier chunks from which they are derived and in the order in which they were generated during information extraction.

Each document chunk corresponds to a *Layer-0* node, and each relation sentence forms a *Layer-1* node. For quick access, each *Layer-1* node connects to its corresponding *Layer-0* nodes—those within the cluster from which it was derived—by establishing directed edges. Additionally, adjacent *Layer-1* nodes (according to their index) are also linked via edges. This indexing step preserves the temporal order of the source document while aligning it with its fine-grained relation units for subsequent graph construction.

3.1.5 Graph Establishing

As shown in Fig. 2, the final graph is established by combining the outputs of the previous stages into a unified node-based structure. Each node stores (1) an index, (2) a text embedding, (3) linked-node information, and (4) its associated text field, which is either a source-chunk text or a relation text extracted from a summary. The index preserves the position of the node within the original document flow, and the linked-node information specifies which other nodes are connected to the current node. These stored node links are then instantiated as graph edges, allowing the graph to encode both hierarchical source–relation connections and local chronological adjacency.

The resulting structure is a two-layer graph in which each layer represents the document at a different granularity. The lower layer consists of the original document chunks arranged in their sequential order, thereby preserving the surface narrative flow of the source document. The upper layer consists of fine-grained relation nodes extracted from the summaries of chunk groups, providing compact semantic units for retrieval. Each upper-layer relation node is linked back to the chunk group from which it was derived, which preserves traceability to the original evidence while also enabling contextual reconstruction during retrieval. By jointly modeling fine-grained semantic facts and their neighboring temporal context, the graph allows ChronoRAG to retrieve precise evidence without losing the narrative coherence of the source text.

3.2 Online Passage Retrieval

At inference time, we handle a query through a hierarchical retrieval process that leverages the constructed graph to assemble a rich, chronologically-aware context for the LLM.

3.2.1 Hierarchical Retrieving

We leverage the hierarchical granularity of *Layer 1* and *Layer 0* for retrieval. We begin by retrieving high-precision relation descriptions from *Layer 1* based on semantic similarity to the query. Then, using the links established during indexing, we retrieve the related *Layer 0* chunks to provide a comprehensive and balanced context. As illustrated in Fig. 3, this process first identifies a key event in *Layer 1* and later retrieves the detailed source text from *Layer 0*. This step is crucial because *Layer 0* often retains omitted details and original dialogues that are valuable for question answering.

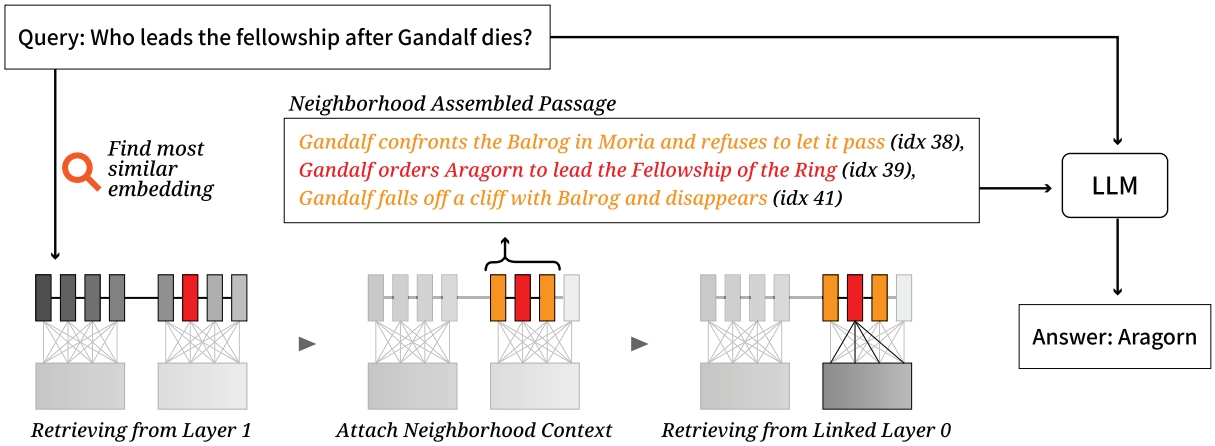

Figure 3: The online passage retrieval process of ChronoRAG for a sample query. This demonstrates how the pre-constructed graph is used at inference time to assemble a chronologically coherent context for the LLM.

3.2.2 Neighborhood Assembling

We then augment retrieved relational descriptions with their surrounding context to reconstruct a narrative flow. Rather than relying on isolated facts, we aim to provide contextually rich information.

As in the example of Fig. 3, after ChronoRAG retrieves a key event, such as "*Gandalf orders Aragorn to lead the Fellowship of the Ring (idx 39)*", the system automatically appends its chronological neighbors, including "*Gandalf confronts the Balrog... (idx 38)*" and "*Gandalf falls off a cliff... (idx 41)*". This creates a coherent, temporally ordered passage that preserves the local storyline, providing crucial context that isolated facts would lack. The overall procedure is summarized in Algorithm 1.

Algorithm 1: Neighborhood assembling

Input: Query q; Layer-1 relation nodes R with temporal index $\mathrm{idx}(\cdot)$;
Top-k retrieved nodes K; neighborhood window size w
Output: Chronologically assembled passage P
$N \leftarrow \varnothing$
foreach $r \in K$ **do**
  foreach $r' \in R$ **do**

(Continued)

Algorithm 1 (continued)

```
        if |idx(r′) − idx(r)| ≤ w then
            N ← N ∪ {r′}
P ← ∅
foreach r ∈ N do
    append text of r to P
return P;
```

3.2.3 Answer Generation

Finally, we combine the original query with the context obtained through hierarchical retrieval and neighborhood assembling and feed them into the language model. We separate each passage by double line breaks and sort by relevance, enabling accurate and coherent answer generation.

4 Experimental Results

In this section, we describe the experimental settings for validating the effectiveness of the proposed method and report the comparison results with five existing methods.

4.1 Experimental Settings

4.1.1 Datasets

We evaluate ChronoRAG on the test splits of NarrativeQA [6] and GutenQA [7], two benchmarks for long-context narrative question answering. NarrativeQA contains 10,557 story–question–answer triples, with source narratives drawn from Project Gutenberg books and movie scripts scraped from the web. GutenQA contains 3000 question–answer pairs built over long-form literary narratives from Project Gutenberg. These benchmarks allow us to evaluate the proposed method on different forms of narrative documents under long-context settings. The average source-document length is about 57,780 tokens for NarrativeQA and about 156,700 tokens for GutenQA

In addition to the full test sets, we construct an auxiliary *temporal-cue subset* for each benchmark by selecting questions that contain at least one explicit temporal cue word, such as *when*, *while*, *during*, *after*, or *before*. This procedure yields 1111 questions from NarrativeQA and 662 from GutenQA. Because these cue words impose explicit temporal conditions on question interpretation, this heuristic provides a simple and reproducible diagnostic subset enriched with questions that are more likely to require chronology-sensitive understanding. We use this subset for auxiliary analysis.

4.1.2 Preprocessing

For both datasets, we preprocess each source document by retaining only the title and main body text, while removing non-content metadata such as author information, copyright notices, and other auxiliary fields. All methods, including ChronoRAG and the comparison baselines, construct their document representations from this preprocessed text and retrieve evidence only from this material during answer generation.

The two datasets differ in how chunk-level inputs are provided. NarrativeQA does not include predefined chunk boundaries, and its source documents are often too long to be directly structured by methods that require chunk-level processing. Therefore, for methods without a dataset-provided chunk structure, we first segment NarrativeQA documents using the same length-based chunking strategy adopted in

RAPTOR [5] and then perform method-specific structuring on top of these chunks. In contrast, GutenQA provides semantically chunked inputs as part of the dataset, and we use those chunks directly as the basis for subsequent structuring and retrieval. This preprocessing ensures that each method operates on tractable input units while preserving a consistent evaluation setting across methods.

4.1.3 Evaluation Metric

We measure answer quality using ROUGE-L [25], which computes the Longest Common Subsequence (LCS) overlap between a generated answer and its corresponding human reference. Due to the short and pronoun-heavy nature of NarrativeQA answers, ROUGE-L effectively captures agreement in key word sequences without penalizing minor rephrasings.

In addition to ROUGE-L, we employ cosine similarity and LLM-based evaluation to better assess semantic fidelity. ROUGE may fail to capture semantically similar answers that differ lexically, so we incorporate complementary metrics to address this limitation. We compute cosine similarity using the Snowflake model [26], which is also used during the retrieval process. It measures the embedding-based similarity between the generated answer and the reference answer.

For LLM-based evaluation, we use GPT-4.1 Mini [27]. Given the question, summary, gold passage, and ground-truth answer, the GPT model performs binary classification—[Correct] or [Wrong]—to determine whether the generated answer can be considered valid. See Appendix A for the detailed prompt.

4.1.4 Baselines

We compare against five existing methods that differ in information extraction, representation, and retrieval structure:

- **NaiveRAG:** A standard RAG pipeline that performs chunk-level retrieval only, without further structuring [4].
- **RAPTOR:** Clusters semantically similar chunks via embedding similarity and builds a recursive summarization tree over clusters to guide retrieval—CT (Collapsed Tree) flattens each root-to-leaf path into one high-level summary, whereas TT (Tree Traversal) retains the full hierarchy and drills down level-by-level to gather finer-grained context [5].
- **LightRAG:** Constructs a lightweight entity–relation graph to enable fast context retrieval using dual-level extraction, prioritizing computational efficiency and incremental updates [18].
- **GraphRAG:** Builds a richer graph with detailed relation weighting and neighborhood assembly to support deeper multi-hop retrieval, capturing both high-level relation summaries and their underlying chunks [17].
- **Propositionizer:** Transforms the entire source text into fine-grained propositions (atomic sentences) and treats each proposition as a retrieval unit, then feeds retrieved propositions into the generation model [12].

4.1.5 Implementation Details

All methods share the same answer generator, embedding model, decoding strategy, and context budget of 1500 tokens. This token budget serves as the primary shared evaluation constraint. We do not enforce a uniform top-k across methods, because the compared architectures differ in chunk granularity, retrieval units, and intermediate retrieval procedures. Instead, each method uses its own internal retrieval parameters under the same final context-budget constraint, and these settings are reported separately as method-specific implementation details. We perform all summarization and entity–relation extraction steps with

meta-llama-3-8B-Instruct [28]. We compute retrieval scores using dense embedding similarity exclusively in order to isolate the effect of word alteration during summarization and information extraction. Specifically, we employ the arctic-Snowflake-embed-l [26] for generating embeddings, and use unifiedqa-v2-t5-3b-1363200 [29] for final answer generation. For each baseline, the method-specific hyperparameters were chosen based on the strongest ROUGE-L performance on subset of NarrativeQA validation split under the shared 1500 token context-budget setting. For clarity, Table 1 summarizes the retrieval mode, retrieval unit, top-k, and shared token limit used for each method. See Appendix B for the detailed hyperparameters.

Table 1: Retrieval settings used in our experiments. All methods share the same final token limit.

Method	Retrieval	Retrieval Unit	Top-k	Token Limit
NaiveRAG		Chunk	20	
RAPTOR_CT		Summary/Chunk	20	
RAPTOR_TT	Dense	Summary/Chunk	20	
Propositionizer	Retrieval	Proposition	100	1500
LightRAG		Entity/Relation/Chunk (hybird mode)	10 per unit	
GraphRAG		Entity/Relation/Chunk (local mode)	10 per unit	
ChronoRAG		Relation/Chunk	15 per unit	

4.2 Main Results

4.2.1 Performance Comparison

Table 2 reports a clear dataset-dependent pattern. On NarrativeQA, ChronoRAG achieves the best performance among all baselines on both ROUGE-L and LLM-based evaluation, for both the full dataset and the Time Question subset. In particular, it obtains ROUGE-L scores of 0.308 on the full set and 0.268 on the Time Question subset, outperforming the strongest baseline, RAPTOR_CT, which reaches 0.298 and 0.262, respectively. The same trend is observed for LLM Eval, where ChronoRAG records the highest accuracy on both subsets (0.257 and 0.195). Although it does not surpass every baseline on cosine similarity, it remains competitive, indicating that its gains are not limited to lexical overlap but extend to answer validity as judged by semantic and LLM-based metrics.

Table 2: QA Performance on NarrativeQA and GutenQA across ROUGE, CosineSim, and LLM Eval metrics. Top performance is bolded, Second best is underlined. The dagger mark at ROUGE and ACC indicates statistical significance at the 5% level.

NarrativeQA						
Metric	**ROUGE**		**CosineSim**		**LLM Eval (ACC)**	
Subset	**Whole Data**	**Time Question**	**Whole Data**	**Time Question**	**Whole Data**	**Time Question**
NaiveRAG	0.255†	0.227†	0.841	0.844	0.183†	0.144†
Propositionizer	0.262†	0.238†	0.846	0.852	0.189†	0.141†
RAPTOR_CT	<u>0.298†</u>	<u>0.262</u>	**0.854**	**0.858**	<u>0.241†</u>	<u>0.178</u>
RAPTOR_TT	0.289†	0.253†	0.851	<u>0.854</u>	0.231†	0.170†
LightRAG	0.240†	0.214†	0.841	0.845	0.182†	0.123†
GraphRAG	0.195†	0.185†	0.823	0.830	0.139†	0.106†

(Continued)

Table 2 (continued)

Metric	ROUGE		CosineSim		LLM Eval (ACC)	
Subset	Whole Data	Time Question	Whole Data	Time Question	Whole Data	Time Question
NarrativeQA						
ChronoRAG (Ours)	**0.308**	**0.268**	0.853	0.854	**0.257**	**0.195**
GutenQA						
NaiveRAG	**0.166**†	**0.172**	0.769	0.778	**0.251**	**0.278**
Propositionizer	0.151†	0.142†	**0.779**	**0.785**	0.167†	0.140†
RAPTOR_CT	0.159	0.164	0.775	0.784	0.244	0.269
RAPTOR_TT	0.104†	0.102†	0.762	0.769	0.086†	0.082†
LightRAG	0.083†	0.083†	0.740	0.750	0.075†	0.077†
GraphRAG	0.122†	0.115†	0.760	0.767	0.134†	0.119†
ChronoRAG (Ours)	0.159	0.170	0.776	**0.785**	0.248	0.275

On GutenQA, the ranking pattern differs. NaiveRAG records the strongest ROUGE-L and LLM Eval scores, with ChronoRAG consistently following as the next-best method. Specifically, NaiveRAG achieves ROUGE-L scores of 0.166 and 0.172 and LLM Eval scores of 0.251 and 0.278 on the full and temporal subsets, respectively, while ChronoRAG reaches 0.159 and 0.170 in ROUGE-L and 0.248 and 0.275 in LLM Eval. Despite this gap, ChronoRAG remains highly competitive across all metrics and stays near the top even under a different question construction regime. Notably, its cosine similarity is also among the strongest on GutenQA, ranking second on the full set and tied for the best score on the Time Question subset. Taken together, these results indicate that ChronoRAG does not suffer a large performance drop across datasets and remains one of the top-ranked methods in both benchmarks.

4.2.2 Discussion

We attribute this difference to the way the two benchmarks are constructed. NarrativeQA questions are created from summaries and paired with free-form answers, which makes broader narrative reconstruction and local event ordering especially useful. In contrast, GutenQA is built directly from original books using automatically generated, highly specific questions, and it provides concise verbatim answer spans anchored in the source text. As a result, GutenQA more strongly rewards direct access to local source wording, which naturally favors chunk-level retrieval methods such as NaiveRAG. ChronoRAG, by contrast, first abstracts chunks into relation descriptions and then assembles adjacent chronological context, which is particularly helpful when answering requires recovering how nearby events connect rather than locating a short source span alone. This helps explain why ChronoRAG is strongest on NarrativeQA while still remaining competitive on GutenQA. The difference in ChronoRAG's relative ranking between NarrativeQA and GutenQA suggests that the method is more beneficial when answering depends on recovering nearby narrative context. Its relative advantage becomes smaller when a single directly matching passage is often sufficient.

4.3 Analysis

4.3.1 Ablation Study

We conduct ablation studies to assess the contribution of each component of ChronoRAG. As shown in Table 3, removing any component leads to a performance drop, indicating that the full framework benefits from combining fine-grained relation-based retrieval with local context expansion. Excluding neighborhood assembly causes a consistent decrease, with a larger drop on the temporal-cue subset, suggesting that adjacent contextual linking is particularly helpful when questions depend more strongly on local chronological cues. Removing chunk summarization also degrades performance, suggesting that shallow summarization helps produce retrieval-friendly relation descriptions by suppressing noisy details. The largest drop occurs when relation extraction is removed, indicating that structured relation descriptions serve as effective retrieval keys. Overall, the ablation results show that ChronoRAG performs best when relation descriptions are used for retrieval and the retrieved units are expanded with adjacent context and linked source chunks.

Table 3: Ablation study of ChronoRAG's core components on the NarrativeQA dataset, with performance measured by ROUGE-L. The dagger mark at ROUGE and ACC indicates statistical significance at the 5% level.

Method	Whole Data	Time Question
ChronoRAG	0.308	0.268
w/o Passage Assembling	0.295†	0.252†
w/o Chunk Summarization	0.272†	0.233†
w/o Relation Extraction	0.255†	0.227†

4.3.2 Trade-Off between Linking Window and the Number of Retrieved Passage

We analyze the trade-off of using a larger linking window for passage assembly. While a wider window provides more local context, it also reduces the number of distinct passages that can be retrieved within a fixed token budget. Our experiment confirms this is detrimental; as shown in Table 4, extending the window to two neighbors ("Extended Link Window") lowers performance on both the whole dataset and the temporal questions subset. This result validates that our default approach of using a more concise, immediately adjacent context is more effective.

Table 4: Analysis of design variations within the ChronoRAG framework on the NarrativeQA dataset measured by ROUGE-L. The dagger mark at ROUGE and ACC indicates statistical significance at the 5% level.

NarrativeQA	Whole Data	Time Question
ChronoRAG	0.308	0.268
Extended Link Window	0.300†	0.258†
Merged Key	0.302†	0.257†

4.3.3 Key-Value Separation in Information Retrieval

We investigate the effectiveness of key-value retrieval design of ChronoRAG, which separates the precise fact used for retrieval (the key) from the broader context provided to the model (the value). To validate this, we test an alternative "Merged Key" approach where the retrieved fact and its neighbors are combined into a single text unit before retrieval. As shown in Table 4, this modification results in a slight performance

decrease, indicating that our key-value separation is an effective strategy for balancing retrieval precision and contextual coherence.

4.3.4 Effect of Neighborhood Assembling

We validate the effectiveness of neighborhood assembling in enriching the retrieved context with information that is chronologically relevant but not necessarily the most semantically similar to the query. To demonstrate this, we compare the average embedding similarity between the query and the initially retrieved sentences vs. the final assembled passages. As shown in Table 5, the average similarity for the assembled passages is discernibly lower than that of the sentences retrieved by similarity alone. This gap indicates that the neighboring passages, while chronologically adjacent, are semantically distinct from the initial query hit. In narrative texts where surrounding content carries strong causal or temporal relevance, this mechanism allows the model to incorporate pertinent information beyond the limits of pure similarity search, thereby improving the context for answer generation.

Table 5: Embedding similarity between query and retrieved/assembled passages.

	# of Sentence	**Mean (Similarity)**
Retrieved Sentences	104,981	0.838
Assembled Sentences	209,092	0.785

4.3.5 Case Study

Fig. 4 presents excerpts of the original passages retrieved by each method for the example shown in Fig. 1. RAPTOR retrieves summary passages, which enable access to content covering a wide range of information. However, these summaries frequently include information that is not pertinent to the query, or conversely, omit critical details necessary for answering the question due to length constraints imposed by the summarization process. LightRAG and GraphRAG extract entities and relations directly from the original text. In particular, GraphRAG was found to underperform compared to direct retrieval to the source chunk, likely due to its tendency to include exhaustive explanations of all elements. Propositionizer and LightRAG offer relatively general-level granularity explanations, yet they still struggle to address questions that require understanding the changes in the relationship between Anna and George. In contrast, ChronoRAG identifies the minimal set of chronologically adjacent passages while suppressing unrelated narrative details, illustrating its strength in maintaining temporal coherence and reducing retrieval noise.

In Table 6, we present another representative case demonstrating how ChronoRAG assembles adjacent passages. Bracketed sentences mark the retrieved evidence aligned with the query, while the surrounding context ensures coherence. This example shows how linking preserves narrative flow and enables the model to answer correctly (“a dog”), reducing ambiguity compared to isolated retrieval.

But ChronoRAG is less effective on queries that require document-level synthesis beyond the local neighborhood window, such as identifying the overall theme of a story. Hierarchical summarization methods such as RAPTOR can better support such queries by aggregating information across the document, whereas ChronoRAG is designed for locally coherent, chronology-preserving retrieval.

Query: Where is George Darrow residing when he prepares to join Anna Leath in France?		
Method	Model Answer	Retrieved Context
(a) ChronoRAG	London	*George Darrow and Anna have a past romantic relationship and are reuniting after 12 years, George Darrow is in London for a dinner party where he reunites with Anna, Darrow and Anna Summers have a past connection and are rekindling their relationship*
(b) RAPTOR	a country estate with his friend Owen.	*The story revolves around George Darrow, a young man who is on leave from his military duties and is staying at a country estate with his friend Owen. Darrow is struggling to come to terms with his feelings for Anna, a woman who...*
(c) LightRAG	athenee	*-----Entities(KG)-----* *Anna is a complex and multifaceted character who is deeply involved in the story. She is the step-mother of Owen....* *-----Relationships(KG)-----* *[{"description": "Anna is concerned about her step-son's departure..."},*
(d) GraphRAG	"DOVER", "GEO"	*-----Entities-----* *0,"RESTAURANT","LOCATION","The restaurant is a location where Anna..."* *1,"DOVER","GEO","Dover is a location where George Darrow takes a train to."* *-----Relationships-----* *0,"CHELSEA","DARROW","Darrow has a past connection with Chelsea...* *1,"ANNA","EFFIE'S EDUCATION","Anna is concerned about Effie's education...*
(e) Propositionizer	Givre	*Anna decided to accompany Darrow to Paris.* *Darrow's disappointment was tempered by the certainty of being with Mrs. Leath again before she left for France.* *Darrow was under the same roof with Anna again.*

Figure 4: A qualitative comparison of retrieved context and model answers for the query, "Where is George Darrow residing when he prepares to join Anna Leath in France?".

Table 6: Representative linking case used in ChronoRAG.

Example Linking Case
Correct: [Correct] Question: What kind of animal does Anna have when Gurov first sees her? Golden Answer: ["A dog", "Small dog"] Generated Answer: a dog Passage: Dmitri Gurov becomes infatuated with the lady with the dog and tries to get to know her, **[Gurov is drawn to Anna's innocence and vulnerability, and they spend time together]**, Gurov and Anna share a romantic moment, but Anna is overcome with guilt and shame. Kovrin and the Pesotskys are preparing for the wedding, **[Kovrin is deeply in love with Tanya, and their relationship is central to the story]**, The black monk appears to Kovrin, filling him with pride and a sense of exalted consequence, and Kovrin is obsessed with his work. Gurov is drawn to Anna's innocence and vulnerability, and they spend time together, **[Gurov and Anna share a romantic moment, but Anna is overcome with guilt and shame]**, Anna confesses her infidelity to her husband, feeling she has betrayed him.

4.3.6 Computation Costs

Our pipeline is computationally efficient, requiring just two LLM calls per 1000 tokens for graph construction. Although this cost increases linearly with document length, it remains lower than competing methods like recursive summarization. Furthermore, only one LLM call is required for answer generation during search, with our method still attaining the highest performance despite its efficiency.

5 Limitations

Our proposed ChronoRAG framework explicitly models temporal order to improve narrative question answering. However, the approach has several limitations. First, while our graph construction pipeline is lightweight compared to prior graph-based methods, it still requires multiple LLM calls for summarization and relation extraction, which may introduce latency in large-scale deployments. Second, our evaluation focuses primarily on two English narrative datasets (NarrativeQA and GutenQA), and results may not directly generalize to non-English narratives or other domains such as legal or medical texts. Third, Although our method improves performance on chronology-sensitive questions, it does not yet capture more complex discourse phenomena such as causal chains spanning distant events. Moreover, our temporal-question subset is constructed heuristically using cue words. As a result, it does not fully capture all questions involving temporal or causal structure. Future work could explore integrating richer discourse structures and extending experiments to multilingual or domain-specific corpora.

6 Conclusion

We present ChronoRAG, an RAG framework that can effectively and efficiently handle narrative text. Our framework refines content through summarization and relation extraction, and improves overall performance through simple passage augmentation that connects adjacent events via an index. This suggests that it is important not only to organize individual events and elements in narrative texts but also to connect events that are spatially and temporally close to each other.

Acknowledgement: Administrative and technical support from Chung-Ang University is gratefully acknowledged.

Funding Statement: This research was supported by Institute for Information & Communications Technology Planning & Evaluation (IITP) through the Korea government (MSIT) under Grant No. 2021-0-01341 (Artificial Intelligence Graduate School Program (Chung-Ang University)) and National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2025-24683575). And this research also was supported by the Chung-Ang University Graduate Research Scholarship in 2026.

Author Contributions: Byeongjeong Kim: Conceptualization, Methodology, Software, Writing—original draft, Writing—review & editing, Validation. Jeonghyun Park: Methodology, Software, Writing—review & editing. Joonho Yang: Software, Methodology. Hwanhee Lee: Supervision, Methodology, Writing—review & editing. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The datasets supporting the findings of this study are publicly available benchmark datasets, NarrativeQA and GutenQA. The code, prompts, and configuration files used in this study will be released in a public repository upon acceptance of the manuscript. Until public release, the code, prompts, and configuration files are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable. This study does not involve human subjects, animal experiments, or any sensitive personal data.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A Prompts

Appendix A.1 Graph Construction Prompt

As shown in Table A1, we employ task-specific prompts for both summarization and entity–relation extraction. For summarization, the model is instructed to condense each cluster of document chunks into a concise description without pronouns, ensuring that the resulting text remains self-contained. For entity–relation extraction, the model is guided by a structured instruction that requires listing entities with type and description, followed by explicit relationships between entities with a numeric strength score. This structured output is essential for constructing the ChronoRAG graph, as it enables us to represent both the factual content of the story and the temporal or relational dependencies between entities. By combining these two prompts, we distill long narrative texts into coherent graph structures that support accurate and temporally consistent retrieval.

Table A1: Prompt templates used in ChronoRAG for summarization and entity–relation extraction.

Summarization Prompt
System: Write a summary of the following context as short as possible within five sentences. DO NOT USE PRONOUN. Context: <*document chunk*> Summary:
Entity–Relation Extraction Prompt
Goal Given a text document that is potentially relevant to this activity and a list of entity types, identify all entities of those types from the text and all relationships among the identified entities.
Steps 1. Identify all entities. For each identified entity, extract: – *entity_name*: Name of the entity, capitalized – *entity_type*: One of *[Leading Role, Supporting Role, Object]* – *entity_description*: Comprehensive description of the entity's attributes and activities *Format: ("entity"/<entity_name>/<entity_type>/<entity_description>)* 2. From step 1 entities, identify clearly related (source, target) pairs and extract: – *source_entity*, *target_entity* – *relationship_description* – *relationship_strength* (numeric) *Format:* *("relationship"/<source_entity>/<target_entity>/<relationship_description>/<relationship_strength>)* 3. Return all items in English as a single list delimited by *&*. 4. Finish with <*End*>.

Appendix A.2 LLM Eval

In addition, as shown in Tables A2 and A3, we assess model outputs with an LLM-based judge. GPT-4.1 Mini is prompted with a structured template that includes the question, a gold passage providing narrative context, the gold answer, and the model-generated answer. The prompt explicitly instructs the judge to output only one of two labels—*[Correct]* or *[Wrong]*—without explanation. This design ensures consistency and

avoids subjective variation. The evaluation complements automatic metrics such as ROUGE and embedding-based similarity by correctly recognizing semantically valid answers even when they differ lexically.

Table A2: Prompt used for LLM-based evaluation on narrativeQA.

System Prompt
You are the grader who determines the correct answer precisely.
User Prompt
Determine whether the user's answer to the following question is correct or not. Choose only one of the two options: *[Correct]* or *[Wrong]*. Do not explain your reasoning; state only your judgment. - *Question: <question>* - *Summary: "<document_summary>"* - *Golden answer: <gold_answer>* - *User's answer: <model_answer>* - *Judgement:*

Table A3: Prompt used for LLM-based evaluation on GutenQA.

System Prompt
You are the grader who determines the correct answer precisely.
User Prompt
Determine whether the user's answer to the following question is correct or not. Choose only one of the two options: *[Correct]* or *[Wrong]*. Do not explain your reasoning; state only your judgment. - *Question: <question>* - *Literature Context: "<chunk_must_contain>"* - *Golden answer: <gold_answer>* - *User's answer: <model_answer>* - *Judgement:*

Appendix B Supplementary Implementation Details

We conduct our experiments using an AMD EPYC 7313 CPU (3.0 GHz) paired with four NVIDIA RTX 4090 GPUs. We use Python 3.11.5 and PyTorch 2.3.1 for the software environment. We access meta-llama-3-8B-Instruct via the OpenRouter (2025) API with temperature set to 0 (greedy decoding) for generating answers from GutenQA. The detailed hyperparameters used in our experiments can be found in Table A4.

We utilize the nano-graphrag repository[1] and the lightrag repository[2] to implement the GraphRAG and LightRAG baselines, respectively. For proposition generation of Propositionizer from the original documents, we employ the chentong00/proposition-izer-wiki-flan-t5-large model.

[1] https://github.com/gusye1234/nano-graphrag
[2] https://github.com/HKUDS/LightRAG

Table A4: Configuration parameters for ChronoRAG pipeline.

Parameter	Value
Max token length in chunk	100
Number of cluster in chunk	10
Max token length in summarization	2000
Max token length in Entity Relation Extraction	2000
do_sample	False
Summarization model	meta-llama/Llama-3.1-8B-Instruct
Entity relation extraction model	meta-llama/Llama-3.1-8B-Instruct
Model for answer generation	unifiedqa-v2-t5-3b-4281363200
Embedding model for text embedding	Snowflake/snowflake-arctic-embed-l
Max token length of context	1500
Retrieving top_k	15/each layer
Max number of retrieved passage	20

In our experiments, GraphRAG is configured in local mode and LightRAG in hybrid mode. For both methods, we first collect candidates by taking the top 10 retrieved units from each of the entity, relation and chunk-level views, and then construct the final context from up to 20 retrieved units, truncating the assembled evidence whenever it exceeds the shared 1500 token budget. In contrast, because Propositionizer yields much finer-grained and shorter textual units, we retrieve up to the top 100 propositions and retain as many as possible within the same 1500 token constraint.

References

1. Pang RY, Parrish A, Joshi N, Nangia N, Phang J, Chen A, et al. QuALITY: question answering with long input texts, yes! In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022; 2022 Jul 10–15; Seattle, WA, USA. p. 5336–58.
2. Liu J, Zhu D, Bai Z, He Y, Liao H, Que H, et al. A comprehensive survey on long context language modeling. arXiv:2503.17407. 2025.
3. Wang C, Duan H, Zhang S, Lin D, Chen K. Ada-LEval: evaluating long-context LLMs with length-adaptable benchmarks. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers); 2024 Jun 16–21; Mexico City, Mexico. p. 3712–24.
4. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. Adv Neural Inf Process Syst. 2020;33:9459–74.
5. Sarthi P, Abdullah S, Tuli A, Khanna S, Goldie A, Manning CD. RAPTOR: recursive abstractive processing for tree-organized retrieval. In: Proceedings of the International Conference on Learning Representations (ICLR); 2024 May 7–11; Vienna, Austria.
6. Kočiskỳ T, Schwarz J, Blunsom P, Dyer C, Hermann KM, Melis G, et al. The NarrativeQA reading comprehension challenge. Trans Assoc Comput Linguist. 2018;6(3):317. doi:10.1162/tacl_a_00023.
7. Duarte AV, Marques JD, Graça M, Freire M, Li L, Oliveira AL. LumberChunker: long-form narrative document segmentation. In: Al-Onaizan Y, Bansal M, Chen YN, editors. Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2024; 2024 Nov 12–16; Miami, FL, USA. Stroudsburg, PA, USA: Association for Computational Linguistics; 2024. p. 6473–86.
8. Caciularu A, Dagan I, Goldberger J, Cohan A. Long context question answering via supervised contrastive learning. In: Carpuat M, de Marneffe MC, Ruiz M IV, editors. Proceedings of the 2022 Conference of the North American

Chapter of the Association for Computational Linguistics: Human Language Technologies; 2022 Jul 10–15; Seattle, WA, USA. Stroudsburg, PA, USA: Association for Computational Linguistics; 2022. p. 2872–9.
9. Qiu J, Liu Z, Liu Z, Murthy R, Zhang J, Chen H, et al. Locobench: a benchmark for long-context large language models in complex software engineering. arXiv:2509.09614. 2025.
10. Beltagy I, Peters ME, Cohan A. Longformer: the long-document transformer. arXiv:2004.05150. 2020.
11. Zaheer M, Guruganesh G, Dubey KA, Ainslie J, Alberti C, Ontanon S, et al. Big bird: transformers for longer sequences. Adv Neural Inf Process Syst. 2020;33:17283–97.
12. Chen T, Wang H, Chen S, Yu W, Ma K, Zhao X, et al. Dense x retrieval: what retrieval granularity should we use? In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing; 2024 Nov 12–16; Miami, FL, USA. p. 15159–77.
13. Gunjal A, Durrett G. Molecular facts: desiderata for decontextualization in LLM fact verification. In: Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2024; 2024 Nov 12–16; Miami, FL, USA. p. 3751–68.
14. Chen H, Pasunuru R, Weston J, Celikyilmaz A. Walking down the memory maze: beyond context limit through interactive reading. arXiv:2310.05029. 2023.
15. Lee KH, Chen X, Furuta H, Canny J, Fischer I. A human-inspired reading agent with gist memory of very long contexts. In: Proceedings of the 41st International Conference on Machine Learning; 2024 Jul 21–27; Vienna, Austria. p. 26396–415.
16. Etzioni O, Banko M, Soderland S, Weld DS. Open information extraction from the web. Commun ACM. 2008;51(12):68–74. doi:10.1145/1409360.1409378.
17. Edge D, Trinh H, Cheng N, Bradley J, Chao A, Mody A, et al. From local to global: a graph rag approach to query-focused summarization. arXiv:2404.16130. 2024.
18. Guo Z, Xia L, Yu Y, Ao T, Huang C. LightRAG: simple and fast retrieval-augmented generation. arXiv:2410.05779. 2024.
19. Yang Z, Wang Y, Shi Z, Yao Y, Liang L, Ding K, et al. EventRAG: enhancing LLM generation with event knowledge graphs. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2025 Jul 27–Aug 1; Vienna, Austria. p. 16967–79.
20. Zhang ZY, Li Z, Li Y, Ding B, Low BKH. Respecting temporal-causal consistency: entity-event knowledge graphs for retrieval-augmented generation. arXiv:2506.05939. 2025.
21. Yang R, Xue H, Razzak I, Hacid H, Salim FD. Beyond single pass, looping through time: kG-IRAG with iterative knowledge retrieval. arXiv:2503.14234. 2025.
22. Yang Z, Qi P, Zhang S, Bengio Y, Cohen W, Salakhutdinov R, et al. HotpotQA: a dataset for diverse, explainable multi-hop question answering. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing; 2018 Oct 31–Nov 4; Brussels, Belgium. p. 2369–80.
23. Baek I, Chang H, Kim B, Lee J, Lee H. Probing-RAG: self-probing to guide language models in selective document retrieval. In: Chiruzzo L, Ritter A, Wang L, editors. Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2025; 2025 May 3–7; Atlanta, GA, USA. Albuquerque, NM, USA: Association for Computational Linguistics; 2025. p. 3287–304.
24. Saxena A, Tripathi A, Talukdar P. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings, Online. In: Jurafsky D, Chai J, Schluter N, Tetreault J, editors. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics; 2020 Jul 5–10; Online. Stroudsburg, PA, USA: Association for Computational Linguistics; 2020. p. 4498–507.
25. Lin CY. Rouge: a package for automatic evaluation of summaries. In: Text summarization branches out. Stroudsburg, PA, USA: Association for Computational Linguistics; 2004. p. 74–81.
26. Merrick L, Xu D, Nuti G, Campos D. Arctic-embed: scalable, efficient, and accurate text embedding models. arXiv:2405.05374. 2024.
27. Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, Aleman FL, et al. GPT-4 technical report. arXiv:2303.08774. 2023.

28. Grattafiori A, Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, et al. The llama 3 herd of models. arXiv:2407.21783. 2024.
29. Khashabi D, Kordi Y, Hajishirzi H. UnifiedQA-v2: stronger generalization via broader cross-format training. arXiv:2202.12359. 2022.