Keystone-Docs RAG at TREC 2025: Coverage-Aware Few-Document Augmentation via Narrative Decomposition

Yukio Uematsu[1], Koyuki Otani[1], Tomoyuki Shiroyama[1], and Masaaki Tsuchida[1]

[1]Department of Information Science, Faculty of Science and Technology, Tokyo University of Science, Noda, Chiba, Japan,
yukio@rs.tus.ac.jp, 6322012@ed.tus.ac.jp, 6322041@ed.tus.ac.jp, masaaki.tsuchida@rs.tus.ac.jp

1 Introduction

The proliferation of large language models (LLMs) with long-context capabilities has enabled Retrieval-Augmented Generation (RAG) [8] to process numerous context segments. However, the introduction of a large number of segments has been reported to degrade RAG's answer generation accuracy [4][9]. Specifically, Jin et.al.[4] report that high-precision retrieval often returns hard negative examples—documents containing similar but differing opinions or facts—which can negatively influence LLM generation by introducing confusion.

The *Narrative* inputs targeted by TREC this year (long texts composed of multiple sentences) require diverse and explanatory responses that account for shifting viewpoints, temporal sequences, and implied relationships, rather than single, fact-oriented answers [3, 6, 10]. We define a Keystone-Doc as a document rich in information covering various aspects of a Narrative. We propose Keystone-Docs RAG, which aims to mitigate the aforementioned problems by selecting a minimal, yet sufficient, set of documents and providing them as RAG context.

Keystone-Docs RAG primarily consists of two phases: the Keystone-Docs derivation phase and the answer generation phase using the derived documents. First, in the derivation phase, multiple viewpoints are extracted from the Narrative to identify components of the Keystone-Docs, and documents comprehensively covering these viewpoints are selected as Keystone-Docs. In the answer generation phase, the answer is fundamentally generated using the derived Keystone-Docs as input. However, we also compare and evaluate a method that derives and integrates answers based on each viewpoint obtained from the Narrative.

We structured Keystone-Docs RAG into the following steps:

Derivation of Candidate Reference Documents via Narrative Decomposition
The multiple viewpoints within the Narrative are broken down using an LLM, and the decomposed viewpoints are converted into Decomposed Queries.

Document Retrieval using Decomposed Queries Documents are retrieved using the Decomposed Queries. This involves hybrid search leveraging both Dense and Sparse retrieval.

Answer Generation Answers are generated from the document candidates retrieved via Decomposed Queries using two methods: Retrieval-level Fusion and Answer-level Fusion.

This paper discusses the Document Retrieval (R Task) conducted to obtain Keystone-Docs in Section 2 and the overall flow of the RAG Task in Section 3.

2 Retrieval Task

This section describes the design of the Retrieval (R) Task in the TREC RAG Track. The central premise of Retrieval in this study is as follows:

> **A Narrative query is composed of multiple viewpoints, events, and constraints, and the "few but necessary and sufficient documents" that satisfy these elements are defined as Keystone-Docs.**

In this research, Keystone-Docs consistent with the Narrative query's structure were identified using a three-stage method:

1. **Narrative Understanding and Decomposition** The long input Narrative is decomposed into multiple queries to identify documents capable of fulfilling the requirements of each query. Documents that comprehensively satisfy these multiple query requirements are positioned as **candidates for Keystone-Docs**.

2. **Hybrid Search and RRF Fusion** For each Decomposed Query, Sparse (BM25) [11] and Dense Retrieval [5] are executed in parallel, and their rankings are fused using RRF (Reciprocal Rank Fusion) [2].

3. **Segment Assignment** To create a Run compliant with the TREC RAG Track format, the segment within the Keystone-Docs that satisfies the requirements of each Decomposed Query is identified, and that segment is submitted as the R Task run.

2.1 Narrative Understanding and Decomposition

Our **Narrative Decomposition** is based on concepts from prior work known as **Query Decomposition (QD)** and **Question Decomposition** [1, 7]. These methods have been shown to be effective in improving retrieval coverage for multi-hop question answering. However, the input for the TREC RAG Track, which is our target, is a Narrative containing multiple perspectives and background information, not merely a single question. Our Narrative Decomposition extends traditional QD not only to split the query into multiple perspectives but also to extract the central topic, thereby generating queries that remain aligned with the original objective. This aims to achieve the goal of "selecting highly readable Keystone-Docs." We performed Narrative Decomposition using the following three steps with an LLM:

1. **Narrative Decomposition**: The long Narrative is split into multiple clear, concise, and single-faceted questions to generate multiple Decomposed Queries.
2. **Narrative Topic Extraction**: A single keyword or phrase present in the original Narrative is extracted to identify the central topic of the Decomposed Queries.
3. **Narrative Topic Expansion**: Search term expansion is performed to include synonyms and related terms from the background text.

By using the generated Decomposed Queries and Narrative Topic, it is possible to retrieve documents by decomposing the required viewpoints for the answer without losing the Narrative's core subject. Ultimately, a total of 771 Decomposed Queries, along with the Narrative_topic and Expanded_Narrative_topic for the 105 Narratives provided in the TREC RAG Track, were generated.

2.1.1 Narrative Decomposition

In Narrative Decomposition, the complex intent contained in the Narrative Narr is understood and transformed into **Decomposed Queries** (simple, specific questions) which are searchable units. Specifically, a set of n_{Narr} Decomposed Queries $\mathcal{Q}(\text{Narr}) = \{q_1, q_2, \ldots, q_{n_{\text{Narr}}}\}$ is generated from Narr.

The System and User prompts used for Narrative Decomposition are shown below. The variable *question* was set to the input Narr.

System Prompt for Narrative Decomposition

```
You are a helpful assistant that takes long and complex
  questions and breaks them down into multiple simple and
  specific questions.
Each output question should be clear, concise, and focused on a
  single aspect of the original input.
```

User Prompt for Narrative Decomposition

```
Break the following complex question into simpler, separate
  questions:

{question}
```

GPT-4o was used as the LLM, and the `temperature` was set to `0` to enhance reproducibility. The generated Decomposed Queries were normalized for case and made unique before use.

2.1.2 Narrative Topic Extraction

Narrative Topic Extraction involves extracting a single key phrase that represents the overall subject of the Narrative. The purpose of this function is to prevent Query Drift during query decomposition, ensuring that the decomposed queries remain focused on the Narrative's subject.

The specific user prompt used for Narrative Topic Extraction is shown below.

User Prompt for Narrative Topic Extraction

```
If you had to extract only one main topic the person wants to
  understand from the following text, what would it be?
Please respond with the shortest possible word or phrase, using
  only expressions that appear in the text.

{narrative}
```

Similar to Narrative Decomposition, GPT-4o (`temperature=0`) was used as the LLM, and the output was a single word or short phrase that most concisely represented the Narrative's subject. This aims to suppress the introduction of external knowledge and enhance consistency with the next step, synonym expansion.

2.1.3 Narrative Topic Expansion

Narrative Topic Expansion generates various paraphrases (synonyms and syntactic variations) for the extracted Narrative Topic. This is intended to improve Recall in subsequent information retrieval.

Proper nouns that appear as examples within the Narrative, which are prone to causing Query drift, were not expanded; instead, other terms were expanded. The model used was GPT-4o, and the generation was expanded to meet a specified number of queries (NUM_QUERIES = 10). The specific prompts used are shown below.

System Prompt for Narrative Topic Expansion

```
You are an expert paraphrasing assistant.
Given a short phrase or sentence, produce exactly {NUM_QUERIES}
  distinct keyword-style variants that preserve the full meaning
  .
Do NOT include any question marks or commas **within** each
  variant.
Vary both key nouns with synonyms and query structures,
e.g.:"
impact of X on "Y, "Y in the age of "X, "influence of X on "Y,
  "how X shapes "Y
Output **only** a JSON array of strings.
```

User Prompt for Narrative Topic Expansion

```
Term or short phrase:
{narrative_topic}
Generate exactly {NUM_QUERIES} keyword-style paraphrases as a
  JSON array.
Each entry should preserve the full original meaning, vary
  structure, and avoid 's excessively.
Do **NOT** put commas or question marks inside entries.
```

Hereafter, the combination of Decomposed query + Narrative Topic + expanded Narrative Topic is simply referred to as Query q.

A concrete example of the results of the three processes described above is shown below.

An Example of Narrative Decomposition and Narrative Topic Extraction/Expansion

Input Narrative: I want a thorough understanding of what makes up a community, including its definitions in various contexts like science and what it means to be a 'civilized community.' I'm also interested in related terms like 'grassroots organizations,' how communities set boundaries and priorities, and their roles in important areas such as preparedness and nation-building.

1. Decomposed Queries:

- What are the different definitions of a community in various contexts, such as science?
- What does it mean to be a 'civilized community'?
- What are grassroots organizations, and how do they relate to communities?
- How do communities set boundaries for themselves?
- How do communities determine and prioritize their goals and needs?
- What roles do communities play in areas like preparedness?
- How do communities contribute to nation-building?

2. Extracted Narrative Topic:

```
community
```

3. Expanded Narrative Topic:

```
"group", "collective", "neighborhood", "social group", "
  fellowship", "society", "association", "public", "population",
   "social circle"
```

2.2 Index Construction for Hybrid Search

This section describes the method for constructing the index for hybrid search. As performing Dense retrieval on the entire corpus is inefficient in terms of cost performance, we extracted only a small subset of documents relevant to the Narrative to perform the Hybrid search. Specifically, we extracted the document set using the following method and built the index for Hybrid search using Opensearch (version 2.19):

1. Retrieve the top 1,000 documents using BM25 for all Queries (771 total) created using the method in Section 2.1.
2. Take the unique set of the retrieved documents (resulting in a document set of 140,561).
3. Obtain embeddings for all documents using OpenAI's `text-embedding-3-small`.

Documents exceeding 8,000 tokens (12,078 total) were summarized using OpenAI's gpt-4.1-mini-2025-04-14.

The specific prompt used for summarization in step 3 is as follows. The variable *target_token* was adjusted to be less than 8,000 tokens.

Prompt for Document summarization

```
Please summarize the following text in English,
keeping it under approximately {target_tokens} tokens
while preserving the important information:

{text}
```

2.3 Retrieval Process for R Task

This section describes the process of retrieving documents containing multi-faceted information for the Narrative using the queries created and the search system built in the previous sections. This process derives Keystone-Docs that cover the entire Narrative through the following two stages of Reciprocal Rank Fusion (RRF).

2.3.1 Step 1: Fusion of Hybrid Search Results per Query

First, for each Decomposed Query q_i, Sparse search (BM25) and Dense Retrieval (cosine similarity) are executed in parallel. The respective result lists are $L_{\text{BM25}}(q_i)$ and $L_{\text{Dense}}(q_i)$, and these are fused using RRF [2] to derive the hybrid search score $s_{\text{hyb}}(d \mid q_i)$ for Query q_i.

$$s_{\text{hyb}}(d \mid q_i) = \frac{1}{k + \text{rank}_{\text{BM25}}(d \mid q_i)} + \frac{1}{k + \text{rank}_{\text{Dense}}(d \mid q_i)}$$

where,

- $\text{rank}_{\text{BM25}}(d \mid q_i)$: The rank of document d in the BM25 search for q_i.
- $\text{rank}_{\text{Dense}}(d \mid q_i)$: The rank of document d in the Dense Retrieval for q_i.
- k: Constant (bias term).

Based on this score, the top $N_{\text{query}} = 1000$ documents R_i are retrieved for each Query q_i. This R_i forms the basis of the input for the subsequent RAG task (Section 3.1).

2.3.2 Step 2: Narrative-wide Document Fusion (Keystone-Docs Derivation)

Next, the search results from all Decomposed Queries derived from the Narrative Narr are fused to create a final ranking for the entire Narrative. Let $\mathcal{Q}(\text{Narr})$ be the set of Decomposed Queries generated from Narrative Narr. The Narrative fusion score $S_{\text{Narr}}(d)$ for document d is calculated by fusing the rank of d in all $q_i \in \mathcal{Q}(\text{Narr})$ using RRF.

$$S_{\text{Narr}}(d) = \sum_{q_i \in \mathcal{Q}(\text{Narr})} \frac{1}{k' + \text{rank}_{\text{hyb}}(d \mid q_i)}$$

where,

- $\text{rank}_{\text{hyb}}(d \mid q_i)$: The rank of document d in the hybrid search (Step 1) for Query q_i.
- k': Constant (bias term) used for Narrative-level fusion.

This fusion prioritizes documents that consistently show high relevance across multiple viewpoints (Decomposed Queries), thus selecting Keystone-Docs at the top. Finally, based on this score $S_{\text{Narr}}(d)$, the top $N_{\text{narr}} = 100$ documents per Narrative are output for the Retrieval Task.

2.4 Segment Assignment

Since Keystone-Docs RAG inherently features document-level selection of Keystone-Docs, the setup differs from the segment-level selection required for the TREC 2025 R Task. We converted the results to the segment level for submission.

Specifically, suppose a document d identified by retrieval is divided into a set of segments $\mathcal{S}_d = \{s_{d,1}, s_{d,2}, \ldots, s_{d,m}\}$. From this set, a single segment s^* that best reflects the content of the Narrative Narr is selected.

Segment selection is performed based on the cosine similarity CosSim between the embedding vector of the entire Narrative E_{Narr} and the embedding vector of each segment $E_{s_{d,j}}$.

$$s^* = \underset{s_{d,j} \in \mathcal{S}_d}{\arg\max} \left(\text{CosSim}(E_{\text{Narr}}, E_{s_{d,j}}) \right)$$

where,

- s^*: The optimal segment selected (the final citation target).
- $\mathcal{S}_d$: The set of segments comprising document d.
- $s_{d,j}$: The j-th segment included in the set $\mathcal{S}_d$.
- E_{Narr}: The embedding vector of the entire Narrative.
- $E_{s_{d,j}}$: The embedding vector of segment $s_{d,j}$.

OpenAI's `text-embedding-3-small` was used as the embedding model.

This process selects the segment with the highest semantic similarity to the overall Narrative, functioning as the representative segment for document d in the Retrieval Task and as the minimum unit for citation after answer generation in the RAG Task.

Table 1: Configuration of Submitted Runs for Retrieval Task

RUN ID	**Decomposition**	**RRF** k	**Fusion Strategy**
`uema2lab_narrative`	×	-	Baseline
`uema2lab_rrf`	○	60	Standard RRF Fusion (High Precision)
`uema2lab_rrf_k10`	○	10	Low-k RRF Fusion (High Coverage)
`uema2lab_segment`	○	-	Two-Stage Fusion

2.5 Submitted Runs for the R Task

To verify the effectiveness of the proposed Narrative Decomposition and RRF fusion strategy, we designed and submitted the four Retrieval Runs shown in Table 1 to the TREC RAG Track. For all Runs, the RRF bias term in the first stage of the hybrid search was commonly set to $k = 60$.

`uema2lab_narrative` (Baseline) This Run serves as a baseline to evaluate the effectiveness of the proposed method. It performs Hybrid search using the entire Narrative as a single query, without Narrative Decomposition. The top 20 documents of the search results are used as the output for the Retrieval Task.

`uema2lab_rrf` (Standard RRF Fusion Model) Run RRF RRF k = 60 This run is a standard RRF-based fusion model designed for comparison, in which the RRF k parameter is set to 60. Under this setting, rank decay is relatively gradual, reducing the relative impact of rank differences within each decomposed query. As a result, documents that are not necessarily highly ranked from any single perspective can still be favorably evaluated if they appear consistently across multiple perspectives.

`uema2lab_rrf_k10` (Diversity Fusion Model) This run is a validation model for the Narrative-Level RRF fusion (Section 2.3, Step 2), in which the RRF k parameter is set to 10. Under this configuration, rank decay is steeper than with $k = 60$, thereby placing stronger emphasis on documents ranked highly within each decomposed query. As a result, documents that consistently achieve high ranks across multiple perspectives (Keystone Docs) are more likely to be positioned at the top, promoting the selection of documents that encompass multifaceted viewpoints.

`uema2lab_segment` (Two-Stage Fusion Model) This Run applies a unique method that integrates documents based on the frequency of document overlap (search frequency) among Queries, without using RRF. By ranking the top 20 documents of the Decomposed Query search results based on their appearance count, it attempts to prioritize documents supported by many viewpoints, ensuring relevance to the overall Narrative. Ties are broken by prioritizing documents with higher RRF scores.

The document set obtained from each Run was finally converted to segment-level results according to the method described in Section 2.3 for submission.

3 RAG Task

This section describes the answer generation (Augmented Generation; AG) using the set of documents obtained for each Decomposed Query from the Retrieval Task. In the Retrieval Task (Section 2), we decomposed the Narrative into multiple Decomposed Queries $\mathcal{Q}(\mathrm{Narr})$ and obtained a set of documents R_i for each query $q_i \in \mathcal{Q}(\mathrm{Narr})$. However, the ultimate goal is to generate a single, comprehensive answer for the entire Narrative. Therefore, a strategy is required to integrate the multiple document sets (or the partial answers based on them) obtained for each Decomposed Query into the final answer.

We implemented the following two answer generation fusion strategies:

1. **Retrieval-level Fusion**: A method where the search results of all Decomposed Queries are merged, re-ranked, and input as a single context to the generative model. Contextual diversity is ensured using the proposed metric, the Parasol Score.

2. **Answer-level Fusion**: A method where individual answers (Sub-answers) are generated for each Decomposed Query and then integrated to create the final answer.

3.1 Input for RAG task

For the sequence of Decomposed Queries $\mathcal{Q}(\mathrm{Narr}) = \{q_1, q_2, \ldots, q_{n_{\mathrm{Narr}}}\}$ generated from the Narrative Narr, we use the top $K = 20$ documents R_i for each q_i. This document set is determined based on the hybrid search score $s_{\mathrm{hyb}}(d \mid q_i)$ derived in Step 1 of Section 2.3.

$$R_i \;=\; \mathrm{TopK}_{d\in\mathcal{D}}\big(s_{\mathrm{hyb}}(d \mid q_i),\, K\big), \qquad K = 20 \tag{1}$$

where,

- $\mathcal{Q}(\mathrm{Narr})$: The set of Decomposed Queries generated from Narrative Narr.
- q_i: The i-th Decomposed Query in the set $\mathcal{Q}(\mathrm{Narr})$.
- R_i: The set of Top-K retrieved documents corresponding to query q_i ($K = 20$).

The defined R_i above is an individual set of evidence documents corresponding to each viewpoint q_i of the Narrative. In Strategy 1, all R_i are merged and input to the AG, while in Strategy 2, each R_i is utilized independently for partial answer generation. $\mathcal{D}$ is the entire document collection, and $s_{\mathrm{hyb}}(d \mid q_i)$ is the hybrid search score derived in Section 2.3 Step 1.

3.2 Strategy 1: Retrieval-level Fusion

In this strategy, before integrating the multiple document sets R_i obtained per Decomposed Query as context input to the LLM, the documents are Reranked using the proposed metric, the Parasol Score, to prioritize documents that are comprehensive and highly relevant to the entire Narrative, and then input as a single context to the LLM.

3.2.1 Reranking using Parasol Score

The Parasol Score is a metric that integrates the UMBRELA scores $u_{q,d} \in \{0, 1, 2, 3\}$, which indicate the relevance between a Decomposed Query and a document, for each Narr. The UMBRELA score is computed using an LLM [12].

The Parasol Score for a document d against a Narrative is defined as:

$$\text{Parasol Score}(\text{Narr}, d) = \sum_{q \in \mathcal{Q}(\text{Narr})} \log(u_{q,d} + 1) \tag{2}$$

where,

- $u_{q,d}$: The UMBRELA score indicating the relevance between Decomposed Query q and document d ($u_{q,d} \in \{0, 1, 2, 3\}$).

This logarithmic summation format is designed to penalize documents with low scores for some queries and to highly evaluate documents (Keystone-Docs) that balance and cover the multifaceted viewpoints constituting the Narrative.

3.2.2 Augmented Generation Pipeline

The top 20 documents ranked by the Parasol Score are formatted into XML to be easily recognizable as structured data by the LLM, and answer generation (AG) is performed using this context and the Narrative as input. We constructed a pipeline using the long-context capable Gemini 2.5 Pro to obtain the final answer through the following main steps:

Augmented Generation The LLM generates a comprehensive answer of 300 to 400 words based on the formatted Top 20 document context and the input where the main argument of the query (Narrative) is emphasized by being presented twice in the prompt.

Answer Summarization This is a post-processing step if the generated answer exceeds the specified word count (400 words). The LLM is strictly prohibited from introducing new facts and performs summarization while maintaining the content consistency to adjust the length.

Reference Assignment The generated answer text and the retrieved documents are input, and the ID of the source document is assigned as evidence to each sentence in the answer.

Note that segment-level citation identification (Segment Assignment), necessary for the final submission format, is executed according to the common method described in Section 3.4 (Segment Assignment).

The prompt used for answer generation is as follows. The Question is included twice in the prompt to prevent the LLM from forgetting instructions by restating the prompt.

Prompt for Answer Generation

```
You are an expert assistant tasked with answering the user's
  question.
Base your answer *solely* on the information provided in the "
  Context Passages" section.
Generate a comprehensive answer text.
The answer *must* be between 300 and 400 words.

Question:{narrative}
====================
Context Passages:{context}
====================
Question:{narrative}
====================
Answer:
```

3.3 Strategy 2: Answer-level Fusion

Answer-level Fusion obtains the final answer by synthesizing Sub-answers generated for each Decomposed Query. The essence of this strategy lies in "Divide and Conquer": instead of solving the entire complex Narrative at once, it aims to generate focused answers for each decomposed viewpoint (Decomposed Query) and integrate them to construct an accurate answer while preventing information loss or hallucination.

3.3.1 Step 1: Sub-answer Generation

The partial answer a_i is generated using each Decomposed Query q_i and its corresponding evidence set R_i.

Ideal Generation Process (Ideal Approach) Ideally, in generating each partial answer, the LLM should be explicitly given the "Decomposed Query q_i" as the question, as shown in the following equation:

$$a_i = \mathrm{Gen}_\theta\Big(\mathrm{Prompt}\big(q_i,\ R_i\big)\Big) \tag{3}$$

This is expected to increase the resolution of each viewpoint, as the LLM can concentrate on extracting and generating only the information necessary for that specific viewpoint (q_i) from R_i, without being influenced by the overall Narrative context.

Actual Implementation However, due to prompt constraints in the implementation of this experiment, we did not explicitly include the Decomposed Query q_i, but instead used the entire Narrative as input.

$$a_i = \mathrm{Gen}_\theta\Big(\mathrm{Prompt}\big(\mathrm{Narr},\ R_i\big)\Big) \tag{4}$$

This setting introduces a bias where the LLM constantly attempts to answer the entire Narrative. However, since the input context R_i is the set of documents (Keystone-Docs candidates) retrieved based on q_i, the generated content ultimately strongly reflects the viewpoint of q_i.

The actual prompt used is as follows. (Note that 'NARRATIVE' is input instead of q_i).

Prompt for Sub-answer Generation (Actual Used)

```
You are an expert assistant. Produce an Augmented Generation (
  RAG) answer for the narrative using only the provided context.
STRICT REQUIREMENTS:
- The final answer must be a JSON object with a single key "
  answer" whose value is an array of sentences.
- Each sentence object must have:
  - "text": the sentence text (English).
  - "citations": an array of ZERO-INDEXED integers that refer to
   the "references" list below.
- The total length of all sentences combined must be **between {
  MIN_WORDS} and {MAX_WORDS} words** (inclusive).
- Cite at least one reference for each sentence. Use multiple
  indices where appropriate.
- Do NOT invent references; use only the indices shown below. Do
   NOT output segment IDs in citations when using indices.

NARRATIVE: {narrative}
REFERENCES: {ref_text}
CONTEXT PASSAGES: {context_blocks}

Return only JSON like:
{{"answer":[{{"text":"...","citations":[0,3]}},{{"text":"...","
  citations":[1]}}]}}"
```

Here, $MIN\_WORDS = 350, MAX\_WORDS = 400$ were set, and the search result R_i was input.

3.3.2 Step 2: Final Synthesis

The resulting sequence of partial answers $\mathcal{A} = \{a_1, \ldots, a_n\}$ is integrated to obtain the final response A_{ans}.

Ideal Integration Process (Ideal Approach) Ideally, in the final step, the LLM should understand the content of the partial answers $\mathcal{A}$ as context and re-generate a coherent and comprehensive answer that aligns with the intent of the original Narrative:

$$A_{\text{ans}} = \text{Gen}_\theta(\text{Prompt}_{\text{synthesis}}, \mathcal{A}, \text{Narr}) \tag{5}$$

This process requires not just simple concatenation, but pruning redundant information and smoothly connecting the contexts based on the Narrative as the main axis.

Actual Implementation However, to maintain the accuracy of the "correspondence between generated sentences and citations" in RAG, and to avoid the risk of hallucination from re-generation, this experiment adopted an "Editing"-based fusion approach. Specifically, the task was formulated as an editing task: inputting the concatenated partial answers and performing redundancy removal and word count adjustment in the context of the Narrative while preserving the citation information.

The actual prompt used is as follows. The Narrative is explicitly provided here as a guideline for editing.

Prompt for Sub-answer Integration (Actual Used)

```
You are a careful editor.
Given the ANSWER_SENTENCES below (a JSON array of objects with
  fields "text" and "citations"),
rewrite ONLY the "text" fields so that the total word count is
  **between {min_w} and {max_w} words (inclusive)**.
CRITICAL RULES:
- Keep the number of sentences the same and preserve their order
  .
- DO NOT modify, add, or remove any "citations". Keep each
  sentence's "citations" array exactly as is.
- Do not introduce new facts. Use fluent academic English. Be
  concise.

Return only the updated JSON array (no extra text, no code
  fences).

ANSWER_SENTENCES:
{ANSWERS}
```

Here, $min\_w = 350$ and $max\_w = 400$ were set. ANSWERS were input as a concatenated JSON array of the partial answers obtained for each q_i.

3.4 Segment Assignment

To obtain the segment-level citation information required for the final output of the RAG task, the cited document ID is converted to the most relevant segment ID based on the cosine similarity between the text embeddings of the generated answer sentences and the source document. This process is executed using the same method as the Segment Assignment (Section 2.4) described in Section 2.

3.5 Submitted Runs for RAG Task

Based on the two proposed fusion strategies (Retrieval-level Fusion and Answer-level Fusion), the following three Runs were submitted to the TREC RAG Track. Gemini 1.5 Pro, known for its long-context processing capabilities, was used as the model for answer generation.

`uema2lab_B4` (Retrieval-level Fusion) This run generates the final answer using a document list consolidated through reranking with our proposed metric, the Parasol Score. It serves as the representative implementation of Strategy 1 (Retrieval-level Fusion).

`uema2lab_base` (Answer-level Fusion 1) This standard run generates partial answers for each Decomposed Query using its top 10 retrieved documents, and then integrates these partial answers into a single final response. It represents the baseline implementation of Strategy 2 (Answer-level Fusion).

`uema2lab_rag_fewdoc` (Answer-level Fusion 2) This run reduces the number of retrieved documents used during sub-answer generation to the top 5, in order to lessen the LLM's context-handling load. The resulting sub-answers are then integrated into the final output. This variant of Strategy 2 is designed to evaluate the extreme case of the Keystone-Docs approach.

4 Conclusion

This study hypothesized that excessive context concatenation in long-context RAG degrades readability and consequently reduces answer performance. Based on this concern, we redefined the role of RAG to select a small but high-quality set of evidence (Keystone-Docs). This design philosophy aligns with the goal of enabling the LLM to reason and summarize coherently without compromising the multiple arguments of the Narrative.

To verify the effectiveness of this Keystone-Docs RAG, we implemented a framework incorporating the following key elements and submitted Runs to the TREC RAG Track:

- **Narrative Decomposition**: Decomposing the long-context query into multiple viewpoints to ensure coverage of evidence documents.
- **Retrieval-level Fusion**: Selecting a minimal set of documents (Keystone-Docs) that balance and cover multifaceted viewpoints through Reranking using the proposed Parasol Score (`uema2lab_B4`, etc.).
- **Answer-level Fusion**: A strategy of generating partial answers for each Decomposed Query and structurally integrating them (`uema2lab_base`, etc.).

Our proposed paradigm of "providing necessary and sufficient context in a readable format" offers a practical and reproducible improvement in input design to maximize the capabilities of long-context LLMs.

Moving forward, based on the evaluation results of the TREC RAG Track, we plan to conduct detailed analysis, particularly on the improvement in retrieval coverage through Narrative Decomposition, the effectiveness of selecting minimal documents using the Parasol Score, and the relative performance difference between Retrieval-level Fusion and Answer-level Fusion.

References

[1] P. J. L. Ammann, J. Golde, and A. Akbik. Question decomposition for retrieval-augmented generation. In J. Zhao, M. Wang, and Z. Liu, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 497–507, Vienna, Austria, July 2025. Association for Computational Linguistics.

[2] G. V. Cormack, C. L. A. Clarke, and S. Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*, 2009.

[3] D. Fried et al. Asqa: Factoid questions meet long-form answers. In *NAACL*, 2022.

[4] B. Jin, J. Yoon, J. Han, and S. O. Arik. Long-context llms meet rag: Overcoming challenges for long inputs in rag, 2024.

[5] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

[6] T. Kočiský, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette. The narrativeqa reading comprehension challenge. In *TACL*, 2018.

[7] D. Lee, A. Park, H. Lee, H. Nam, and Y. Maeng. Typed-RAG: Type-aware decomposition of non-factoid questions for retrieval-augmented generation. In H. Fei, K. Tu, Y. Zhang, X. Hu, W. Han, Z. Jia, Z. Zheng, Y. Cao, M. Zhang, W. Lu, N. Siddharth, L. Øvrelid, N. Xue, and Y. Zhang, editors, *Proceedings of the 1st Joint Workshop on Large Language Models and Structure Modeling (XLLM 2025)*, pages 129–152, Vienna, Austria, Aug. 2025. Association for Computational Linguistics.

[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, P. Kuksa, S. Min, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp. In *NeurIPS*, 2020.

[9] N. F. Liu, K. Lin, J. Hewitt, B. D. Paranjape, M. Bevilacqua, P. Liang, and T. B. Hashimoto. Lost in the middle: How language models use long contexts. In *EMNLP*, 2023.

[10] S. Min, J. Michael, H. Hajishirzi, and L. Zettlemoyer. Ambigqa: Answering ambiguous open-domain questions. In *EMNLP*, 2020.

[11] S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. Okapi at TREC3. In *Text REtrieval Conference*, pages 21–30, 1992.

[12] S. Upadhyay, R. Pradeep, N. Thakur, N. Craswell, and J. Lin. Umbrela: Umbrela is the (open-source reproduction of the) bing relevance assessor, 2024.