Improving Long-Context Summarization with Multi-Granularity Retrieval Optimization

Xueyu Chen[1], **Kaitao Song**[2], **Zifan Song**[1], **Dongsheng Li**[2], **Cairong Zhao**[1*]

[1]Department of Computer Science and Technology, Tongji University, China
[2]Microsoft Research Asia, China
{2310901, 2111139, zhaocairong}@tongji.edu.cn, kt.song@njust.edu.cn, dongsli@microsoft.com

Abstract

Retrieval-Augmented Generation (RAG) is an effective solution to overcome the limitations of Large Language Models (LLMs) in terms of specific-domain knowledge and timely information updates. However, current RAG methods typically respond to queries based on isolated segments, lacking the ability to integrate information within the same document. This undermines performance in real-world tasks requiring coherent understanding across an entire document. Notably, the human brain naturally integrates and summarizes prior knowledge upon reading a given text, progressively formulating a comprehensive understanding. Motivated by this cognitive process, we propose the Hierarchical Two-Stage Summarization-based Information Retrieval (HTSIR) method, which preprocesses the corpus prior to retrieval, summarizes continuous texts to obtain integrated information, and constructs a retrieval tree with varying summary granularities. The retrieved information is then processed by a Reranker based on the current question to serve as a context for LLMs. Additionally, as single-step summarization is often imprecise in query-based summarization tasks, we further apply a Refinement module, allowing LLMs to reflect and revise their output to achieve the final result. By combining HTSIR with GPT-4o mini, we achieve state-of-the-art results on complex question tasks across four long-text datasets (NarrativeQA, QASPER, QuALITY, and QMSum), achieving an improvement of about 6 points on the Question Answering (QA) task in QuALITY-HRAD.

Introduction

Large Language Models (LLMs) have made significant contributions and achieved notable advancements in the field of generative AI. Despite their remarkable performance, LLMs still face challenges in specialized domains and adapting to the latest knowledge (Chang et al. 2024; Wang et al. 2025). Retrieval-Augmented Generation (RAG) integrates generative models with information retrieval techniques (Fan et al. 2024), effectively overcoming these limitations of LLMs.

Existing LLMs are constrained by context length limitations, hindering their ability to process excessively long inputs in a single pass. Even within these limitations, performance tends to degrade with longer inputs (Liu et al. 2024b; Zhou et al. 2025). Therefore, the quality of the information recovered as a context for LLMs is critical (Ru et al. 2024). Recognizing that selecting the appropriate retrieval granularity is a straightforward yet effective strategy, Chen et al. designed a new retrieval granularity (Chen et al. 2023) called Propositions, defined as atomic expressions in the text, presented in a concise, self-contained natural language format, with an average of 42 propositions per page of text. Although effective, this method may demand significant computational resources and complex pre-processing. The RAPTOR (Sarthi et al. 2024) retrieval algorithm clusters text fragments by grouping those with similar topics and constructs a retrieval tree to summarize fragments related to the same topic. However, it fails to consider that contextual information is often more coherent and reflects the inherent logic. Moreover, certain types of articles, such as academic papers or technical reports, usually have clear sectional divisions and logical hierarchies. Such structures not only help readers better understand the content but also assist authors in organizing and expressing complex research ideas. Even in long texts without explicit hierarchical structures, such as novels and essays, narrative coherence and artistic expression are often emphasized, while implicit logical hierarchies remain present. In this paper, we treat the hierarchical structure of the article itself as essential information and construct a retrieval corpus with varying levels of granularity, significantly enhancing retrieval performance.

Figure 1(a) shows that most existing methods only retrieve smaller and shorter text segments, while (b) highlights how our approach utilizes text structure and summarization techniques for information retrieval. This paper aims to improve retrieval quality by employing advanced text segmentation and summarization generation techniques. Specifically, we propose the Hierarchical Two-stage Summarization-based Information Retrieval (HTSIR) method, which leverages the hierarchical structure of documents as important information and constructs a retrieval corpus with varying granularity levels to capture the text's inherent logic and structure. By generating summaries of multiple text segments and further processing these summaries, we construct a tree structure that integrates information from lower to higher levels. This approach not only accommodates diverse retrieval granularities, but also significantly improves the accuracy of information retrieval and

*Corresponding author.

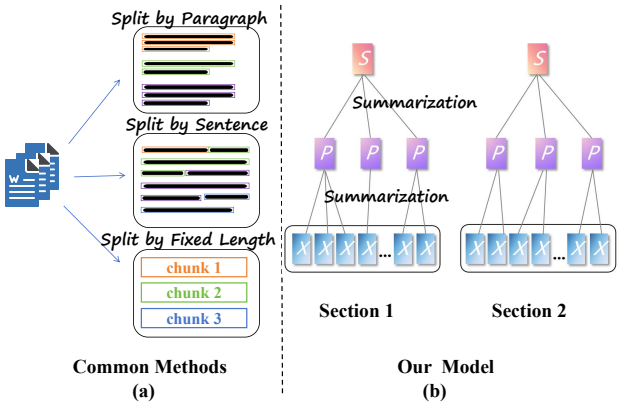

Figure 1: Comparison of text segmentation methods: (a) common model vs. (b) our model.

summarization by capturing the complete hierarchical structure of the text. In the summary generation phase, Refinement module can be selectively applied to leverage feedback from LLMs, producing summaries tailored to diverse needs.

To achieve fast response in large-scale searches, vector retrieval techniques are widely used. This process encodes text into vector representations and employs metrics such as cosine similarity to measure the similarity between the text and the query vector. However, this vectorization inevitably leads to information loss (Singh et al. 2022). Furthermore, the inability to foresee queries before they are received results in a lack of understanding of the query context, further exacerbating the issue of information loss. Therefore, we propose a two-stage retrieval process integrated with a reranking mechanism that is triggered when a user query is submitted. This enables query-specific document analysis, rather than generating a generalized average meaning, thus improving the system's ability to retrieve relevant information. We demonstrate notable improvements in retrieval performance through comprehensive experiments on four long-text datasets for knowledge-intensive questions.

Related Works

Retrieval Methods. In information retrieval, research on retrieval methods remains central. (Shen et al. 2018; Li et al. 2025). Traditional retrieval methods mainly include Boolean retrieval, the Vector Space Model (VSM), and probabilistic models such as BM25 (Jones, Walker, and Robertson 2000). These methods typically rely on lexical matching and statistical properties, offering the advantages of simple computation and intuitive understanding. However, they exhibit limitations when processing queries that require higher-level semantic understanding. Deep learning models in Dense Retrieval map both queries and documents into a continuous vector space, enhancing retrieval efficiency and accuracy (Zhao et al. 2024; Chen et al. 2025). For example, the Bidirectional Encoder Representations from Transformers (BERT) model, through pre-training and fine-tuning techniques, can capture complex semantic relationships in the text, enhancing retrieval performance (Kenton and Toutanova 2019). This approach typically uses Dual Encoders to separately encode queries and documents into vector representations, with relevance assessed via vector similarity. Dense Retrieval better processes semantic information in natural language, particularly excelling in scenarios involving long texts and complex queries.

Reranking Techniques. Reranking techniques refine initial retrieval results to improve effectiveness (Hambarde and Proenca 2023; Mortaheb et al. 2025). The system first uses vector search to filter candidate documents, then reranks them with a sophisticated model for higher-quality results. HTSIR adopts such a two-stage approach, which reranks the retrieved contexts, not only reducing computational burden and improving response speed but also enhancing the ability to handle complex queries. Reranking techniques are divided into Pointwise, Pairwise, and Listwise methods based on different optimization goals and methods (Liu et al. 2009). The Pointwise method sorts documents by predicting the relevance score of individual documents (Cossock and Zhang 2006; Long et al. 2025), the Pairwise method sorts by comparing the relevance between pairs of documents (Boudin 2018; Wu et al. 2025), while the Listwise method directly optimizes the ranking quality of the entire document list (Yang et al. 2025). In recent years, deep learning (Huang et al. 2013; Guo et al. 2020) and pre-trained language models (Fan et al. 2022) (such as BERT) have also been applied to reranking tasks, boosting retrieval performance. LLMs exhibit strong natural language processing capabilities and are increasingly applied to reranking tasks. Through clever prompt design, they can better achieve ranking and perform exceptionally well in zero-shot or few-shot scenarios (Qin et al. 2024; Chen, Pradeep, and Lin 2025).

Retrieval-Augmented Generation. RAG retrieves relevant information from external resources to provide supplementary knowledge to LLMs, enabling them to learn beyond their inherent parameters and mitigating the "lost in the middle" phenomenon (Liu et al. 2024b). This approach enhances the performance of LLMs in long-text tasks (Xu et al. 2023). However, since the retrieved information is not always reliable, SELF-RAG introduces special reflection markers during the generation process, enabling self-assessment of both retrieved and generated content. This method supports on-demand retrieval and self-reflection, enhancing the response quality and factual accuracy of the language model (Asai et al. 2023). Wang et al. proposed an iterative training framework that leverages a reward model based on LLMs feedback to evaluate the quality of candidate examples and then trains a dense retriever based on dual encoders using knowledge distillation (Wang, Yang, and Wei 2024). Xu et al. further improved the quality of retrieval results by optimizing re-ranking and truncation tasks in information retrieval (Xu et al. 2024). Kim et al. introduced a method called Summarizing Retrievals (SuRe) (Kim et al. 2024), which first uses LLMs to generate multiple candidate answers. That said, this process is not well-suited for domain-specific information retrieval and question answering tasks. Subsequently, the method summarizes the retrieved content to enhance the efficiency of information verification. Even so, these RAG methods have

a significant limitation: they struggle to integrate information across multiple passages, as the retrieval process typically treats passages independently. This constraint hampers their ability to achieve a holistic understanding of the information. Although several graph-based RAG methods (e.g., HippoRAG, GraphRAG) have been proposed to address this issue their practical deployment remains constrained by the prohibitively high computational costs of knowledge graph construction—requiring extensive entity-relation extraction, and resource-demanding embedding pretraining (Jimenez Gutierrez et al. 2024; Gutiérrez et al. 2025; Larson and Truitt 2024). In contrast, our proposed tree-based method processes raw text data directly, achieving not only a significant reduction in computational resource consumption but also demonstrating competitive performance.

The HTSIR Retrieval Framework

We define the document set to be retrieved as $D = \{d_1, d_2, \ldots, d_n\}$, where each d_i represents a complete article. Our goal is to answer knowledge-intensive questions Q from D. The overview of HTSIR is illustrated in Figure 2, comprises four key stages: HTSIR Tree Construction, HTSIR Search, Reranking, and Response Generation. Summary generation serves as a critical step in tree construction. We offer an optional optimization approach that leverages LLMs feedback for adaptive refinement, allowing flexible improvement of summaries as needed.

HTSIR Tree Construction

The HTSIR tree consists of three types of nodes: Root Node $\tilde{s}_j$, Intermediate Node p_k, and Leaf Node x_i^j. This hierarchical structure preserves the contextual and semantic consistency of the text within each node and provides information at varying levels of granularity. Primarily considering two different types of articles: one type includes academic papers, technical reports, etc., which inherently exhibit a hierarchical structure with clear chapter divisions and logical organization. The other type includes novels and essays that typically lack a clear hierarchical structure. The process is described in Algorithms 1.

Text Data with Hierarchical Structure. Each document d_i consists of various sections $S = \{s_1, s_2, \ldots, s_m\}$. Each section s_j represents a distinct part or chapter of the document, such as the Introduction, Methods, Results, etc. Since each section provides independent and specific insights, we integrate section-specific knowledge into the retrieval process. However, supplying LLMs with the full content of one or more sections would create an excessively long context, and potentially leading to the "lost in the middle" issue. Therefore, our approach is to summarize the text of each section $\tilde{s}_j$, thereby retaining a more coarse-grained representation of the information. These section summaries are generated by LLMs.

To obtain fine-grained information, following the paradigm of RAG techniques, we divide s_j into short and contiguous segments of length l, obtaining x_i^j as uniformly sized chunks. Note that if a sentence exceeds the limit of l tokens, we move the entire sentence to the next chunk instead of splitting it in the middle. P_k is the text summarized from r chunks of x_i^j to obtain intermediate-grained information. To avoid mixing information from other topics, these r chunks of x_i^j originate from the same section.

Text Data Without a Clear Hierarchical Structure. For a document d_i without a clear hierarchical structure, we can manually divide it into different sections. By providing simple prompts to LLMs, we can segment the text into several sections that share similar themes. To save costs, when dealing with a dataset containing similar content, we can select a few representative articles. Assuming an average text length of 5000 words and that LLMs identify five themes within the text, we can divide the text into sections of 1000 words each, resulting in five sections. Similarly, we summarize these sections individually to obtain $\tilde{s}_j$.

We then divide d_i into short continuous texts of length l, resulting in x_i^j as chunks of fixed size. P_k is the text summarized from r chunks of x_i^j. Finally, the summarized text, the divided chunks, and the summaries of these chunks will all be included in the corpus to be retrieved.

Algorithm 1: Construction of Hierarchical Document Retrieval Tree

Require: Hierarchical document d_i with sections $S = \{s_1, s_2, \ldots, s_m\}$, hyperparameters l (chunk length) and r (number of chunks for summarization).

Ensure: $T = \{X, P, S\}$ - the hierarchical tree structure, where X is the set of leaf nodes, P is the set of intermediate nodes, and S is the set of root nodes.

```
Initialize an empty tree T.
for each section s_j ∈ S do
    Summarize the text of s_j to create the root node s̃_j.
    Add s̃_j to S.
    Divide s_j into continuous text chunks X_j = {x_1^j, x_2^j, ..., x_n^j}, where each chunk x_k^j has length l.
    for each chunk x_k^j ∈ X_j do
        Add x_k^j as a leaf node to X.
    end for
    Group r consecutive chunks from X_j into a set G = {g_1, g_2, ..., g_⌊n/r⌋}.
    for each group g_p ∈ G do
        Summarize the text of g_p to create an intermediate node p_p.
        Add p_p to P, connected to s̃_j and the corresponding leaf nodes in g_p.
    end for
end for
return X, P, S
```

Collapsed Tree Search Strategy

The importance of nodes and the number of nodes required for retrieval vary significantly across different levels of the tree. Nodes at higher levels (e.g., root or intermediate levels) typically represent broader concepts and may require fewer retrievals, while nodes at deeper levels (e.g., leaf nodes) tend to capture more specific details and may re-

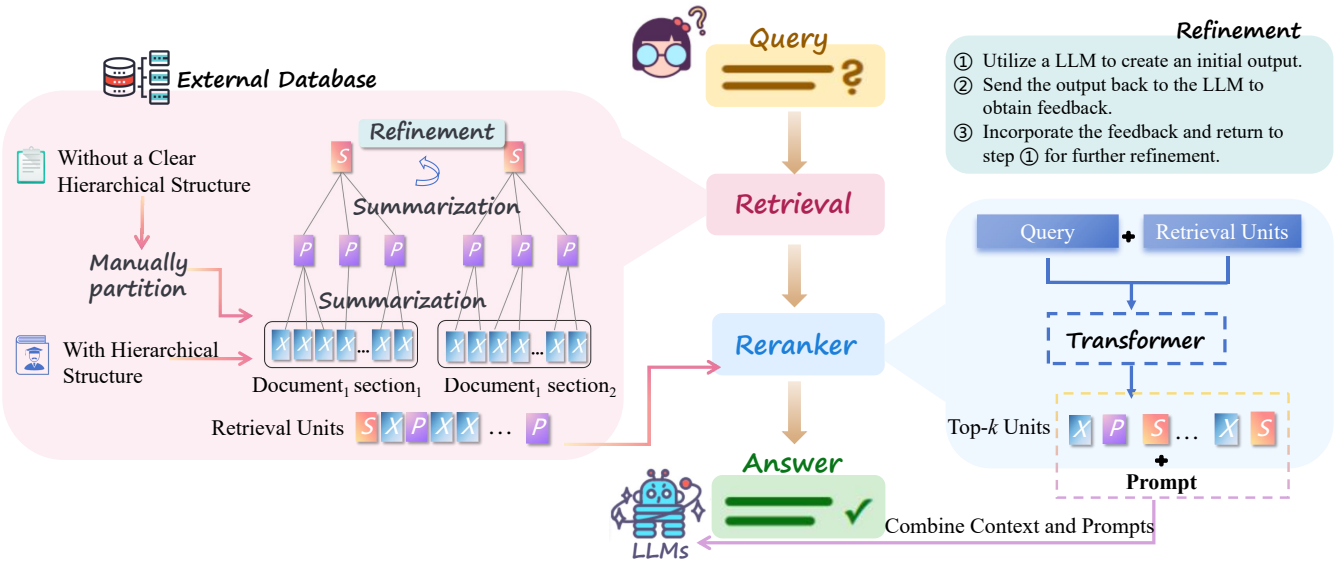

Figure 2: Overview of the HTSIR framework: a retrieval tree is constructed using summarization. Segments are extracted and ranked by semantic similarity to serve as LLMs context. The Refinement module is optional.

quire more retrievals to ensure comprehensive coverage. To achieve this, we introduce the collapsed tree strategy. By aggregating nodes from all levels into a single collapsed set C, this method enables the comparison of nodes across different levels within a unified framework. This approach eliminates the need to predefine the number of nodes to retrieve at each level, allowing the algorithm to dynamically select the most relevant nodes based on their cosine similarity with the query vector.

Compared to the traditional tree traversal method, which processes nodes incrementally at each level (retrieving a fixed number of nodes per level), the collapsed tree strategy improves retrieval accuracy (Sarthi et al. 2024). Additionally, the collapsed tree strategy achieves higher computational efficiency. In practical implementation, there is no need to explicitly construct the tree structure; instead, all nodes can be placed into a single set, enabling batch computation of cosine similarities.

Reranking and Response Generation

Based on the cosine similarity scores computed in the previous step, we extract a subset of candidate nodes from the collapsed set C, which are the most relevant to the query. These candidate nodes form the input for a Reranker model that further refines their order by reassessing the semantic alignment between the query and the candidate texts. Specifically, we utilize widely used reranking models (e.g., BGE-Reranker and Cohere) to perform this refinement. The reranking process ensures that nodes with the highest relevance to the query are prioritized. Finally, we integrate the content of the Top-k nodes and incorporate it as part of the LLMs prompt to generate the final answer to the question.

Feedback-Driven Refinement

In most cases, a single summary of the text is sufficient to effectively improve the performance of the model. However, in some QA tasks focused primarily on summarization, more attention needs to be paid to the details of the article. Therefore, the content of the summary can be further optimized to better meet the requirements. Inspired by self-refine (Madaan et al. 2024), we designed a feedback loop to refine the content of the summary. This optimization process does not require additional training; it simply involves returning the output obtained along with the corresponding feedback to LLMs. The LLMs refine the summary based on the feedback, and this process is repeated until the requirements are satisfied.

Experience

Experiment Settings

We selected several datasets specifically designed for long-text QA tasks: QASPER (Dasigi et al. 2021a), QuALITY (Bowman et al. 2022), NarrativeQA (Kočiskỳ et al. 2018), and QMSum (Zhong et al. 2021). Unless otherwise specified, k set to 10 in this paper. The experiments can be run on a consumer-grade computer, and we will open-source the code after the paper is accepted.

In practical applications, the value of r is relatively easy to determine. Taking the QASPER dataset as an example, which consists of hierarchical text, the average text length is approximately 5,000 words. The root node is constructed by summarizing the content of each individual section in the document (e.g., sections within a research paper), where the text length of each section is generally no more than 500 words. Therefore, l is set to 100, and r is set to 2. For sections shorter than 200 words, to avoid redundancy

Model	ROUGE-L	BLEU-1
BM25 w/o	26.52	21.59
BM25 w + HTSIR	**30.47**	**27.27**
SBert w/o	32.68	23.91
SBert w + HTSIR	**36.43**	**28.66**
DPR w/o	31.07	23.53
DPR w + HTSIR	**34.20**	**29.76**

Table 1: Comparison of the performance of retrieval methods (SBERT, BM25, DPR) with and without HTSIR on the NarrativeQA dataset. The HTSIR model surpasses the baselines for all retrieval methods, achieving higher ROUGE-L and BLEU-1 scores.

and resource waste, we perform summarization only once when constructing the root node. Furthermore, for the Abstract sections in the QASPER dataset, which already serve as comprehensive document summaries, no further summarization is implemented.

In cases without hierarchical information, the QUALITY dataset is an example, where the average text length is also around 5,000 words. The root node is generated from a summary of the text with a length of 1,000 words, while leaf nodes are 100 words in length. Since the values of both the leaf nodes and the root nodes can be easily determined, reasonable values for r, which governs the intermediate nodes, are 2, 3, 4, or 5.

We evaluate HTSIR across multiple LLMs, including both closed-source (GPT-3.5-Turbo, GPT-4o mini) and open-source (Llama3-8B). Our results demonstrate robust performance improvements across different architectures.

Results Analysis

Comparative Baseline Evaluation. We evaluated the performance of the model with and without HTSIR framework, utilizing three embedding models: BM25 (Jones, Walker, and Robertson 2000), SBERT (Reimers and Gurevych 2019), and Dense Passage Retrieval (DPR) (Karpukhin et al. 2020). [1] Our evaluation encompassed three datasets: NarrativeQA, QuALITY, and QASPER. W/o represents the performance of the retrieval models without the hierarchical tree structure.

As shown in Tables 1 and 2, the results indicate that HTSIR consistently outperforms its respective retrievers across all datasets when combined with any retrieval model. This finding further validates the ability of the HTSIR model in answer generation tasks. The HTSIR method integrates information of varying granularity through summarization, enhancing the accuracy and relevance of the final responses.

Contrasting with Advanced Systems. Recent studies have explored the use of graph-based retrieval structures for RAG (Jimenez Gutierrez et al. 2024; Gutiérrez et al. 2025; Larson and Truitt 2024), making it necessary to compare with these methods. These graph-based approaches first require the construction of knowledge graphs. (Jimenez Gutierrez et al. 2024) employed Llama-3-70B-Instruct for NER and GPT-4-mini as the QA reader. To ensure fair comparison, we adopted their reported results and experimental setup (using the NarrativeQA dataset) without reimplementing the knowledge graph. As shown in Table 3, HTSIR achieved a F1 score of 37.4%, surpassing the 25.2% score of HippoRAG 2 and +16.5 points higher than the GraphRAG method. The tree structure we proposed aims to enhance the performance of text retrieval. In contrast, graph-based approaches heavily rely on knowledge bases, requiring documents to be effectively converted into entities and relations, a process that entails significant computational and memory costs.

On QASPER, our method uses GPT-4o mini as the baseline in Table 4. HTSIR achieved a 36.9% Answer F1 score, outperforming VCC (30.8%), a method for long-sequence Transformers. And it improved by 0.2% compared to RAPTOR, which is another tree-based RAG method. Compared to RAPTOR, which requires topic clustering of text blocks, our method focuses more on the logical hierarchy of the text itself. Moreover, we analyze the effectiveness of us-

[1] Summary generation and QA tasks both utilize the GPT-4o mini language model, noted for its affordability and outstanding performance in natural language understanding tasks.

Model	QuALITY Accuracy	QASPER Answer F1
BM25 w/o	51.83	26.56
BM25 w + HTSIR	**57.34**	**28.36**
SBert w/o	55.25	32.65
SBert w + HTSIR	**61.61**	**34.48**
DPR w/o	55.31	32.69
DPR w + HTSIR	**62.57**	**36.95**

Table 2: Performance comparison of Accuracy and Answer F1 for various models (BM25, SBERT, DPR) with and without HTSIR on the QuAlITY-HARD and QASPER dataset. The HTSIR model consistently outperforms the baselines of each respective model.

Model	EM	F1
GraphRAG (Larson and Truitt 2024)	5.4	20.9
HippoRAG (Jimenez Gutierrez et al. 2024)	2.1	16.1
HippoRAG 2 (Gutiérrez et al. 2025)	5.8	25.2
HTSIR	**6.1**	**37.4**

Table 3: Comparison of EM and F1 scores for different graph-based approaches on the NarrativeQA dataset.

Model	Answer F1
LED-base + FullText (Dasigi et al. 2021b)	29.0
LongRAG (Jiang, Ma, and Chen 2024)	26.3
VCC (Zeng et al. 2024)	30.8
RAPTOR (Sarthi et al. 2024)	36.7
GPT-4o mini	31.5
HTSIR	**36.9**

Table 4: Answer F1 scores of various models on the QASPER dataset.

Model	Answer F1
PEAR Llama3-8B (Tan et al. 2025)	18.0
DePaC Llama3-8B (Ma et al. 2024)	17.6
SCPT Llama2-7B (Liu et al. 2024a)	19.7
Llama3-8B LoRA	21.7
HTSIR Llama3-8B LoRA	23.5

Table 5: Comparison of Answer F1 scores for various Llama-based models on the QASPER dataset.

Model	Accuracy
DPR and DeBERTaV3 (Bowman et al. 2022)	46.1
CoLISA (DeBERTaV3) (Dong et al. 2023)	54.7
VCC (Zeng et al. 2024)	56.0
RAPTOR (Sarthi et al. 2024)	56.6
GPT-4o mini	72.1
HTSIR	62.5
HTSIR + GPT-4o mini	**74.0**

Table 6: Accuracy scores of various models on the QuALITY-HARD dataset.

ing open-source and smaller-scale LLMs as baselines. In the summary generation phase, we employed the original pre-trained model to retain its general language capabilities, while in the QA phase, we utilized the model fine-tuned with LoRA to enhance task specificity. Compared to directly using Llama3-8B LoRA as a baseline, our framework achieved a two percentage point improvement in performance. The result in Table 5 demonstrates that our approach HTSIR is applicable not only to closed-source LLMs but also to open-source small models, although the latter performs slightly worse than the former.

Table 6 presents the accuracy scores of various models in the QuALITY-HARD dataset. HTSIR + GPT-4o mini means that the retrieved content from HTSIR is added at the start of the input, followed by the full text, forming the combined input for GPT-4o mini. By leveraging DPR, the HTSIR model achieves higher accuracy compared to the decoding-enhanced DeBERTaV3 model (Bowman et al. 2022). The strengths of HTSIR are especially evident in hard questions, which necessitate longer response times and a deeper comprehension of the articles. Additionally, experiments demonstrate that, within the context window limitations of LLMs, placing the key information retrieved by HTSIR as a supplement at the beginning of the input can further improve the accuracy of the model's responses.

In the QMSum dataset, we compared UL2, which demonstrates versatility and multi-tasking capabilities; and Bart-SLED, which processes long texts using overlapping chunks. Our comparison primarily focused on the results of the ROUGE-L metric to reflect the coherence and completeness of the summaries. As shown in Table 7, the HTSIR model achieved excellent performance with a score of 23.8, higher than the scores of the other comparison models.

Ablation Study on Parameter r. In Section 4.1, we discussed the process of determining the value of parameter r. Table 8 illustrates that the method maintains stable performance across a reasonable range of r values, as identified through a quick manual evaluation. This observation highlights the robustness of the approach.

Model	ROUGE-L
UL2 (Tay et al. 2023)	20.4
BART–SLED (Ivgi, Shaham, and Berant 2023)	23.3
RAPTOR (Sarthi et al. 2024)	22.1
GPT-4o mini	20.8
HTSIR	**23.8**

Table 7: ROUGE-L scores of various models on the QMSum dataset.

r	QuALITY-EASY	QuALITY-HARD
2	83.61	62.48
3	83.74	62.74
4	**84.17**	**62.99**
5	83.91	62.68

Table 8: Effect of the value of r on Accuracy for QuALITY-EASY and QuALITY-HARD datasets.

The Impact of the Refinement Module. We compared the impact of the Refinement module in the QMSum-Committee dataset. In this Refinement module, our goal is to encourage the LLM to pay more attention to the opinions of each participant and important details. It is evident that, whether addressing general or summarization questions, our tree-structured retrieval method can produce relatively reliable answers. DPR+Refinement often provides the most comprehensive descriptions. For example, in response to the question, "How did the National Police Chiefs' Council help in out-of-court disposals?" we provided detailed descriptions of the specific measures taken by the Council, including clear referral guidelines, enhanced collaboration with the Crown Prosecution Service, and increasing transparency and accountability through local oversight groups. We believe that by incorporating the Refinement module, the LLM can achieve feedback and self-improvement based on different settings, thereby better meeting specific needs.

Contribution analysis of the tree structure. To demonstrate performance improvements based on retrieval unit granularity, this section analyzes the contribution of each node layer to evaluating model performance with HTSIR tree retrieval alone. In the QuALITY dataset, we constructed HTSIR trees for each story. Table 9 presents the results for handling hard and easy questions.

In the row labeled "1 layer", the results for nodes containing only x, only p, and only s are presented sequentially from left to right. It is clear that the p node, which has undergone a single summarization, contains the most comprehensive information. The row labeled "2 layers" shows the results of simultaneously retrieving x and s nodes, x and p nodes, and p and s nodes. Compared to the scenario with only one type of node, the retrieval performance declines. Both leaf nodes and intermediate nodes exhibit similar partial information and fundamentally lack global context. We propose that this decline is attributable to the introduction

Start Layer	QuALITY-HRAD			QuALITY-EASY		
	Layer 0	**Layer 1**	**Layer 2**	**Layer 0**	**Layer 1**	**Layer 2**
1 layer	60.57 (x)	61.54 (p)	54.81 (s)	79.31 (x)	81.61 (y)	70.11 (s)
2 layers	57.69 (x+s)	52.88 (p+x)	53.85 (s+p)	73.56 (x+s)	79.31 (y+x)	75.86 (s+y)
3 layers	-	-	**62.50** (all)	-	-	**83.91** (all)

Table 9: Performance of HTSIR tree when querying different tree layers for QuALITY-HRAD and QuALITY-EASY datasets. Columns indicate the initial query layers (highest layers), rows represent different numbers of layers queried.

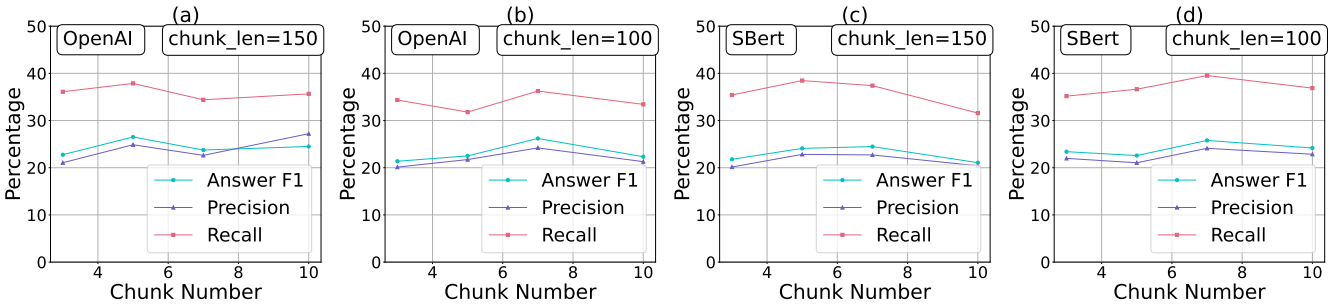

Figure 3: Comparison of different text segmentation methods on the QASPER dataset, evaluating F1 score, Precision, and Recall across varying chunk numbers with OpenAI and SBert embeddings at different chunk lengths.

of redundant information between the two types of nodes, which amplifies the impact of partial or misleading summary information. However, when all three types of nodes are included, the retrieval performance reaches its optimal level. The superior performance with three types of nodes is primarily due to their inclusion of global information. The three nodes provide complementary information: the x node offers specific details, the p node provides a summary, and the s node potentially offers global information, enabling the retrieval system to capture information at multiple levels of granularity.

Evaluation of Text Lengths in Summary Generation. To further analyze the information density of the text, we compared summary generation performance when selecting texts of varying lengths. In this section, we do not construct a tree structure; instead, the corpus consists solely of fixed length text segments x and the corresponding summarized content generated from varying amounts of x.

Figure 3 illustrates the performance metrics—Answer F1, Precision and Recall—of OpenAI and SBert embeddings in the QASPER dataset, with respect to different lengths of text segmentation and the varying numbers of fragments used for summarization. GPT-3.5-Turbo was used for both summarization and QA tasks. The results demonstrate that OpenAI embeddings consistently outperform SBert embeddings, particularly in terms of precision and recall; however, SBert embeddings exhibit relatively stable performance across different chunk counts. A key observation is that peak performance occurred with chunk lengths of 150 (for 5 chunks) and 100 (for 7 chunks), totaling approximately 750 and 700 tokens, respectively. Performance declined when the text length deviated from this range, highlighting the importance of selecting an appropriate length for effective summarization.

Although this approach performs slightly worse than HTSIR—which uses section-based segmentation and multi-level summaries—it provides a simpler alternative by eliminating the need for tree construction. In contrast, HTSIR reduces interference through section-wise content grouping and enriches information via hierarchical summaries. Therefore, if high retrieval accuracy is not pivotal, this method can be used to determine an optimal text length without requiring tree construction.

Conclusion

In this paper, we introduce HTSIR, a method that utilizes the structural characteristics of text to perform summarization at different levels of granularity. This approach constructs a retrieval tree that integrates contextual information from the text. In the first stage, we obtain a subset of candidate nodes from the collapsed set C that are semantically relevant to the initial query through similarity comparisons. The Reranker module then reorders these nodes and selects the top-k candidates as context for LLMs. Optionally, the Refinement module can be flexibly integrated into either the summarization or QA pipeline to accommodate specific requirements. Extensive experiments on four benchmark datasets demonstrate that HTSIR significantly outperforms existing RAG methods and state-of-the-art approaches, particularly in domain-specific QA tasks involving long documents, while effectively handling summarization-based question answering. And HTSIR demonstrates effectiveness across both closed-source and open-source LLMs.

Acknowledgments

This work was supported by National Natural Science Fund of China (No. U25A20527, 62473286).

References

Asai, A.; Wu, Z.; Wang, Y.; Sil, A.; and Hajishirzi, H. 2023. Self-rag: Self-reflective retrieval augmented generation. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Boudin, F. 2018. Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprint arXiv:1803.08721*.

Bowman, S. R.; Chen, A.; He, H.; Joshi, N.; Ma, J.; Nangia, N.; Padmakumar, V.; Pang, R. Y.; Parrish, A.; Phang, J.; et al. 2022. QuALITY: Question Answering with Long Input Texts, Yes! *NAACL 2022*.

Chang, Y.; Wang, X.; Wang, J.; Wu, Y.; Yang, L.; Zhu, K.; Chen, H.; Yi, X.; Wang, C.; Wang, Y.; et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3): 1–45.

Chen, T.; Wang, H.; Chen, S.; Yu, W.; Ma, K.; Zhao, X.; Yu, D.; and Zhang, H. 2023. Dense x retrieval: What retrieval granularity should we use? *arXiv preprint arXiv:2312.06648*.

Chen, X.; Miao, R.; Hu, L.; Zhang, Q.; Song, K.; Naseem, U.; and Zhao, C. 2025. Beyond Topology-based Graph Mining: Deep Analysis Research Networks via Evolutionary Topology and Content Fusion. *Information Fusion*, 103922.

Chen, Z.; Pradeep, R.; and Lin, J. 2025. Accelerating Listwise Reranking: Reproducing and Enhancing FIRST. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 3165–3172.

Cossock, D.; and Zhang, T. 2006. Subset ranking using regression. In *Learning Theory: 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006. Proceedings 19*, 605–619. Springer.

Dasigi, P.; Lo, K.; Beltagy, I.; Cohan, A.; Smith, N. A.; and Gardner, M. 2021a. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*.

Dasigi, P.; Lo, K.; Beltagy, I.; Cohan, A.; Smith, N. A.; and Gardner, M. 2021b. A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers. In Toutanova, K.; Rumshisky, A.; Zettlemoyer, L.; Hakkani-Tur, D.; Beltagy, I.; Bethard, S.; Cotterell, R.; Chakraborty, T.; and Zhou, Y., eds., *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4599–4610. Online: Association for Computational Linguistics.

Dong, M.; Zou, B.; Li, Y.; and Hong, Y. 2023. CoLISA: inner interaction via contrastive learning for multi-choice reading comprehension. In *European Conference on Information Retrieval*, 264–278. Springer.

Fan, W.; Ding, Y.; Ning, L.; Wang, S.; Li, H.; Yin, D.; Chua, T.-S.; and Li, Q. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 6491–6501.

Fan, Y.; Xie, X.; Cai, Y.; Chen, J.; Ma, X.; Li, X.; Zhang, R.; Guo, J.; et al. 2022. Pre-training methods in information retrieval. *Foundations and Trends® in Information Retrieval*, 16(3): 178–317.

Guo, J.; Fan, Y.; Pang, L.; Yang, L.; Ai, Q.; Zamani, H.; Wu, C.; Croft, W. B.; and Cheng, X. 2020. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, 57(6): 102067.

Gutiérrez, B. J.; Shu, Y.; Qi, W.; Zhou, S.; and Su, Y. 2025. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*.

Hambarde, K. A.; and Proenca, H. 2023. Information retrieval: recent advances and beyond. *IEEE Access*.

Huang, P.-S.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2333–2338.

Ivgi, M.; Shaham, U.; and Berant, J. 2023. Efficient long-text understanding with short-text models. *Transactions of the Association for Computational Linguistics*, 11: 284–299.

Jiang, Z.; Ma, X.; and Chen, W. 2024. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*.

Jimenez Gutierrez, B.; Shu, Y.; Gu, Y.; Yasunaga, M.; and Su, Y. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37: 59532–59569.

Jones, K. S.; Walker, S.; and Robertson, S. E. 2000. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information processing & management*, 36(6): 809–840.

Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; and Yih, W.-t. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6769–6781. Online: Association for Computational Linguistics.

Kenton, J. D. M.-W. C.; and Toutanova, L. K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, 2. Minneapolis, Minnesota.

Kim, J.; Nam, J.; Mo, S.; Park, J.; Lee, S.-W.; Seo, M.; Ha, J.-W.; and Shin, J. 2024. SuRe: Summarizing Retrievals using Answer Candidates for Open-domain QA of LLMs. *arXiv preprint arXiv:2404.13081*.

Kočiskỳ, T.; Schwarz, J.; Blunsom, P.; Dyer, C.; Hermann, K. M.; Melis, G.; and Grefenstette, E. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6: 317–328.

Larson, J.; and Truitt, S. 2024. GraphRAG: Unlocking LLM discovery on narrative private data. *Microsoft Research Blog*.

Li, X.; Jin, J.; Zhou, Y.; Zhang, Y.; Zhang, P.; Zhu, Y.; and Dou, Z. 2025. From matching to generation: A survey on generative information retrieval. *ACM Transactions on Information Systems*, 43(3): 1–62.

Liu, K.; Chen, Z.; Fu, Z.; Zhang, W.; Jiang, R.; Zhou, F.; Chen, Y.; Wu, Y.; and Ye, J. 2024a. Structure-aware Domain Knowledge Injection for Large Language Models. *arXiv preprint arXiv:2407.16724*.

Liu, N. F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; and Liang, P. 2024b. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12: 157–173.

Liu, T.-Y.; et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3): 225–331.

Long, K.; Li, S.; Xu, C.; Tang, J.; and Wang, T. 2025. Precise Zero-Shot Pointwise Ranking with LLMs through Post-Aggregated Global Context Information. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2384–2394.

Ma, Z.; An, S.; Lin, Z.; Zou, Y.; Lou, J.-G.; and Xie, B. 2024. Dehallucinating Parallel Context Extension for Retrieval-Augmented Generation. *arXiv preprint arXiv:2412.14905*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Mortaheb, M.; Khojastepour, M. A. A.; Chakradhar, S. T.; and Ulukus, S. 2025. Re-ranking the context for multimodal retrieval augmented generation. *arXiv preprint arXiv:2501.04695*.

Qin, Z.; Jagerman, R.; Hui, K.; Zhuang, H.; Wu, J.; Yan, L.; Shen, J.; Liu, T.; Liu, J.; Metzler, D.; et al. 2024. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, 1504–1518.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Conference on Empirical Methods in Natural Language Processing*.

Ru, D.; Qiu, L.; Hu, X.; Zhang, T.; Shi, P.; Chang, S.; Cheng, J.; Wang, C.; Sun, S.; Li, H.; et al. 2024. RAGChecker: A Fine-grained Framework for Diagnosing Retrieval-Augmented Generation. *arXiv preprint arXiv:2408.08067*.

Sarthi, P.; Abdullah, S.; Tuli, A.; Khanna, S.; Goldie, A.; and Manning, C. D. 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. In *International Conference on Learning Representations (ICLR)*.

Shen, F.; Xu, Y.; Liu, L.; Yang, Y.; Huang, Z.; and Shen, H. T. 2018. Unsupervised Deep Hashing with Similarity-Adaptive and Discrete Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12): 3034–3044.

Singh, K. N.; Devi, S. D.; Devi, H. M.; and Mahanta, A. K. 2022. A novel approach for dimension reduction using word embedding: An enhanced text classification approach. *International Journal of Information Management Data Insights*, 2(1): 100061.

Tan, T.; Qian, Y.; Lv, A.; Lin, H.; Wu, S.; Wang, Y.; Wang, F.; Wu, J.; Lu, X.; and Yan, R. 2025. PEAR: Position-Embedding-Agnostic Attention Re-weighting Enhances Retrieval-Augmented Generation with Zero Inference Overhead. In *Proceedings of the ACM on Web Conference 2025*, 1693–1702.

Tay, Y.; Dehghani, M.; Tran, V. Q.; Garcia, X.; Wei, J.; Wang, X.; Chung, H. W.; Bahri, D.; Schuster, T.; Zheng, S.; Zhou, D.; Houlsby, N.; and Metzler, D. 2023. UL2: Unifying Language Learning Paradigms. In *The Eleventh International Conference on Learning Representations*.

Wang, L.; Yang, N.; and Wei, F. 2024. Learning to Retrieve In-Context Examples for Large Language Models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1752–1767.

Wang, M.; Stoll, A.; Lange, L.; Adel, H.; Schütze, H.; and Strötgen, J. 2025. Bring your own knowledge: A survey of methods for llm knowledge expansion. *arXiv preprint arXiv:2502.12598*.

Wu, J.; Yan, L.; Qin, Z.; Zhuang, H.; Liu, T.; Dong, Z.; Wang, X.; Oosterhuis, H.; et al. 2025. Harnessing Pairwise Ranking Prompting Through Sample-Efficient Ranking Distillation. *arXiv preprint arXiv:2507.04820*.

Xu, P.; Ping, W.; Wu, X.; McAfee, L.; Zhu, C.; Liu, Z.; Subramanian, S.; Bakhturina, E.; Shoeybi, M.; and Catanzaro, B. 2023. Retrieval meets long context large language models. In *The Twelfth International Conference on Learning Representations*.

Xu, S.; Pang, L.; Xu, J.; Shen, H.; and Cheng, X. 2024. List-aware reranking-truncation joint model for search and retrieval-augmented generation. In *Proceedings of the ACM on Web Conference 2024*, 1330–1340.

Yang, E.; Yates, A.; Ricci, K.; Weller, O.; Chari, V.; Van Durme, B.; and Lawrie, D. 2025. Rank-k: Test-time reasoning for listwise reranking. *arXiv preprint arXiv:2505.14432*.

Zeng, Z.; Hawkins, C.; Hong, M.; Zhang, A.; Pappas, N.; Singh, V.; and Zheng, S. 2024. VCC: scaling transformers to 128K tokens or more by prioritizing important tokens. *Advances in Neural Information Processing Systems*, 36.

Zhao, W. X.; Liu, J.; Ren, R.; and Wen, J.-R. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 42(4): 1–60.

Zhong, M.; Yin, D.; Yu, T.; Zaidi, A.; Mutuma, M.; Jha, R.; Awadallah, A. H.; Celikyilmaz, A.; Liu, Y.; Qiu, X.; et al. 2021. QMSum: A new benchmark for query-based multi-domain meeting summarization. *arXiv preprint arXiv:2104.05938*.

Zhou, Y.; Liu, H.; Chen, Z.; Tian, Y.; and Chen, B. 2025. GSM-Infinite: How Do Your LLMs Behave over Infinitely Increasing Context Length and Reasoning Complexity? *arXiv preprint arXiv:2502.05252*.