DTCRS: Dynamic Tree Construction for Recursive Summarization

Guanran Luo[*], **Zhongquan Jian**[*], **Wentao Qiu** , **Meihong Wang, Qingqiang Wu**
School of Informatics, Xiamen University
luoguanran@stu.xmu.edu.cn, wuqq@xmu.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) mitigates the hallucination issues of large language models (LLMs) by integrating external knowledge. For abstractive questions involving multi-step reasoning, knowledge from multiple sections is often required. To address this issue, recent research has introduced recursive summarization, which constructs a hierarchical summary tree by clustering text chunks, integrating information from various parts of the document to provide evidence for abstractive questions. However, summary trees often contain a large number of redundant summary nodes, which not only increase construction time but may also negatively impact question answering. Moreover, recursive summarization is not suitable for all types of questions. We introduce DTCRS, a method that dynamically generates summary trees based on document structure and query semantics. DTCRS determines whether a summary tree is necessary by analyzing the question type. It then decomposes the question and uses the embeddings of sub-questions as initial cluster centers, reducing redundant summaries while improving the relevance between summaries and the question. Our approach significantly reduces summary tree construction time and achieves substantial improvements across three QA tasks. Additionally, we investigate the applicability of recursive summarization to different question types, providing valuable insights for future research.

1 Introduction

Although Large Language Models (LLMs) have demonstrated tremendous advantages across various tasks, updating model knowledge to adapt to the ever-changing world remains a critical issue (Zhang et al., 2023). Compared to methods of fine-tuning models, the Retrieval-Augmented Generation (RAG) paradigm, which combines LLMs with knowledge bases by injecting external knowledge, can update knowledge without modifying model parameters (Wang et al., 2024). RAG has become an important method for mitigating large model hallucinations and enhancing answer interpretability (Li et al., 2024).

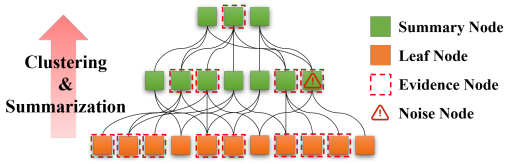

Figure 1: The presence of numerous query-irrelevant summary nodes in the summary tree increases construction time and may adversely affect correct answers.

Most current RAG research only retrieves consecutive short text chunks, which lack sufficient context to answer questions that require integrating information from multiple parts of a document. Recursive summarization (Wang et al., 2023) addresses this issue by hierarchically clustering and summarizing dispersed text chunks to construct a summary tree, thereby integrating the scattered information. Wu et al. propose a method for recursively summarizing entire fiction novels using human feedback (Wu et al., 2021). HIBRIDS introduces hierarchical biases into the Transformer architecture to better capture document structure for long-document summarization (Cao and Wang, 2022). RAPTOR (Sarthi et al., 2024) establishes a new baseline for RAG by constructing hierarchical summary trees for information retrieval, outperforming existing retrieval methods across various QA datasets.

Recursive summarization is effective because it generates summaries at different granularity levels, providing more information for answer generation. However, as shown in Figure 1, we find that it also introduces a large amount of redundant summaries unrelated to the answer, which not only increases

[*]Equal contribution.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10948–10963

computational overhead but may also negatively impact the correctness of the final answer. One key reason for this issue is that traditional summary tree is a static tree based on documents, which splits the original document into text chunks and then generates summaries based on clustered chunks (Sarthi et al., 2024). As a result, the summary tree only reflects the document itself rather than capturing the semantics of the query, leading to an abundance of redundant summaries that are unrelated to the query.

Moreover, whether all types of questions can benefit from recursive summarization remains an unresolved issue. Intuitively, recursive summarization integrates dispersed knowledge within a document, making it particularly beneficial for abstractive questions that require multi-step reasoning. However, for simpler tasks such as extractive or boolean questions, whether recursive summarization provides any advantage remains uncertain (Zhang et al., 2022). Therefore, indiscriminately using recursive summarization is likely to interfere with the generation of answers.

To address the above issues, we propose a Dynamic Tree Construction for Recursive Summarization (DTCRS), which replaces the static summary tree generated solely from the document with a dynamically constructed summary tree based on the document structure and query semantics. This approach enhances the relevance between the summary topics and the query while reducing redundant summaries. Specifically, we first determine the question type. For complex questions that require summarizing information from multiple parts of the document, we generate a table of contents (ToC) for the document. Based on this ToC, we decompose the question into multiple simpler sub-questions. DTCRS then uses the number of sub-questions and their embeddings as the number of clusters and initial cluster centers to perform Gaussian Mixture Model (GMM) clustering on text chunks and generate corresponding text summaries. This process is repeated recursively to construct a summary tree for RAG.

We conduct comprehensive experiments on three QA datasets, and the results show significant improvements over the baselines, indicating that the dynamic summary tree generates more relevant summaries for the questions. We further enhance answer quality by assessing question types before building the summary tree, thereby reducing time overhead. Additionally, we analyze the performance of DTCRA across various types of questions. The experimental results validate our hypothesis that introducing recursive summarization significantly enhances the performance of LLMs on more challenging abstractive questions. However, for extractive and boolean questions, recursive summarization provides no advantage and may even have a negative impact.

2 Related Work

Effectively utilizing long-range context remains a persistent challenge in retrieval-augmented QA. Dai et al. (Dai, 2019) introduced Transformer-XL to overcome fixed-length context limitations; however, recent studies by Sun et al. (Sun et al., 2021) and Liu et al. (Liu et al., 2024b) highlight that contemporary language models still encounter difficulties in fully leveraging extended contexts. To address these challenges, research has explored several complementary directions, which we group here into four major areas.

Retrieval-Augmented Generation. RAG improves QA performance by incorporating external knowledge into model inference. Techniques such as self-reflective prompting, adaptive complexity control, and iterative refinement have proven effective in improving query formulation (Asai et al., 2023; Jeong et al., 2024; Chan et al., 2024). These approaches tend to work well within narrow domains but face difficulties generalizing to more complex or diverse inputs (Zhang et al., 2024; Siriwardhana et al., 2023). By introducing a summarization structure that absorbs and organizes retrieved content, our method aims to maintain the advantages of RAG while reducing its reliance on brittle query-retrieval dependencies.

Retrieval Methods. Open-domain QA has benefited significantly from improvements in retrieval, especially with the rise of dense representations and neural scoring. Early works based on term frequency (Sparck Jones, 1972) have evolved into embedding-based retrieval (Khattab and Zaharia, 2020; Guu et al., 2020; Karpukhin et al., 2020; Liu et al., 2021) that offers better contextual alignment. More recent methods adopt hierarchical or multi-stage retrieval schemes (Arivazhagan et al., 2023), and knowledge distillation has helped balance efficiency and effectiveness (Izacard and Grave, 2020; Roberts et al., 2020; Min et al., 2021). While most of these techniques aim to improve recall and ranking quality, they are typically designed as indepen-

dent modules. In contrast, our strategy connects retrieval output directly to a downstream structure, improving continuity and interpretability.

Text Summarization. Summarization plays a key role in condensing long or multi-document inputs. Recent approaches have explored recursive summarization to extend memory (Wang et al., 2023) and query-focused methods to improve relevance (Deng et al., 2023; Xu and Lapata, 2020). Chunk-level and weakly supervised summarization (Angelidis and Lapata, 2018; Gao et al., 2023; Sarthi et al., 2024; Wu et al., 2021) further increase scalability. These techniques primarily focus on fluency or informativeness, often at the expense of structural clarity. By organizing summaries within a tree-based layout aligned to document sections, we aim to retain coherence while supporting targeted reasoning.

Query Enhancement. Improving the clarity and utility of user questions has become a central technique in QA pipelines. Query rewriting has been shown to improve retrieval for under-specified or ambiguous queries (Mo et al., 2024; Peng et al., 2024), while decomposition supports multi-hop inference (Reppert et al., 2023; Ye et al., 2023; Radhakrishnan et al., 2023). These strategies typically operate as pre-processing steps. In contrast, our approach grounds sub-question generation in the structure of the Summary Tree, using the natural organization of documents to inform and constrain the reasoning process.

Together, these lines of work provide a strong foundation. By drawing on their strengths and embedding them within a unified, structured representation, our method aims to enhance long-context QA in a more interpretable and modular fashion.

3 DTCRS

The overall process of DTCRS is shown in Figure 2. In this section, we first introduce how to classify the questions, then explain how DTCRS handles different types of questions, with a focus on how DTCRS constructs dynamic recursive summary trees for abstractive questions. Finally, we introduce two retrieval methods for the summary tree.

3.1 Question Type Classification

Indiscriminately introducing recursive summarization adds extra overhead and may negatively impact the performance of question answering. Therefore, it is necessary to determine the question type before constructing the recursive tree in order to decide whether recursive summarization is required. We use LLM as a table of contents (ToC) generator and classifier. The ToC c is generated based on the document d, and then c and the original question q are input into the classifier. It outputs a binary label:

$$y = f_{\text{LLM}}(q, c) \in \{0, 1\} \tag{1}$$

The classification criteria are whether the question is complex and whether it requires summarizing information from multiple sections of the ToC to provide a complete answer. If the classifier returns 1, a dynamic summary tree is constructed for that question; otherwise, we use the DPR(Karpukhin et al., 2020) method to retrieve the top k text chunks:

$$S(d, q) = \text{Top}_k\left(\{\text{sim}(q, T_i) \mid T_i \in d\}\right) \tag{2}$$

Where T_i represents the i-th text chunk in document d. $\text{sim}(q, T_i)$ computes the similarity between query q and text chunk T_i.

We provide the prompts for table of contents generation and question classification in Appendix A.

3.2 Dynamic Summary Trees

Our summary trees are dynamic because they generate different summary trees for different questions. There are two main issues that dynamic summary trees address: (1) the redundancy of nodes in the summary tree, and (2) the irrelevance of generated summaries to the query. To address these issues, we describe the construction process of dynamic summary trees below.

Quesion Decomposition. To address the redundancy issue in summary trees, we can reduce the number of clusters at each layer. Previous methods have employed the Bayesian Information Criterion (BIC) (Neath and Cavanaugh, 2012) to determine the optimal number of clusters for the model:

$$\text{BIC} = \ln(n)\, p - 2\ln(\hat{L}), \tag{3}$$

where n is the number of data points, p is the number of parameters in the model, and $\hat{L}$ is the maximized value of the likelihood function of the model.

The number of clusters is determined by the number of text chunks and the number of model parameters. Therefore, the longer the text and the more model parameters, the more redundant nodes may be generated. Assume that the document d

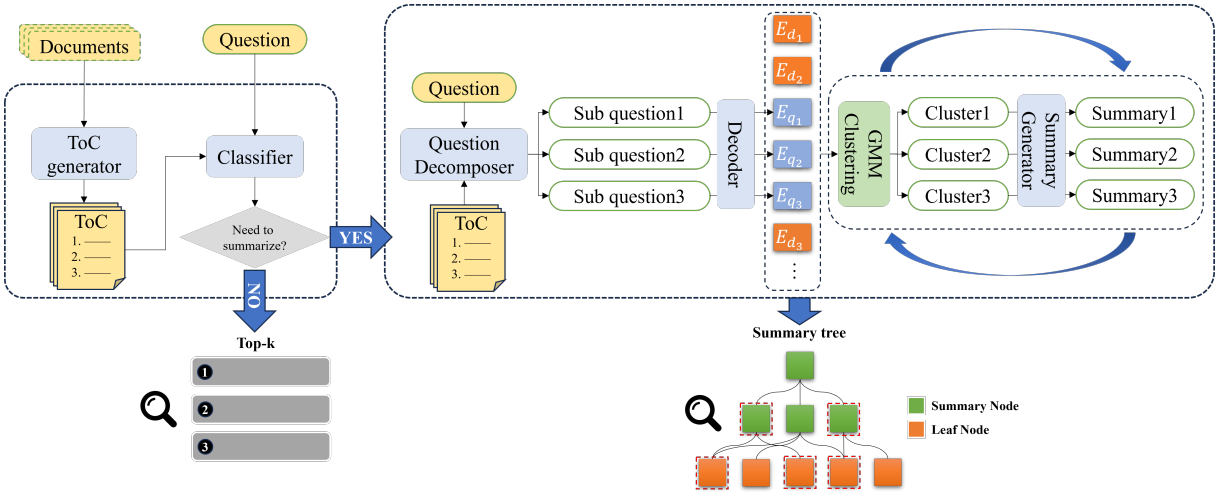

Figure 2: The overall process of DTCRS. The classifier first determines the question type. If summarization of document information is required, a dynamic summary tree is constructed; otherwise, DPR (Karpukhin et al., 2020) is used for retrieval. During the construction of the dynamic summary tree, the question is decomposed based on the document table of contents, with sub-questions serving as initial cluster centers for recursive clustering to generate summaries at different levels.

is divided into N_d chunks, the total computational effort can be described by the geometric series:

$$N_d + \frac{N_d}{2} + \frac{N_d}{4} + \frac{N_d}{8} + \cdots \tag{4}$$

The total computational workload is $2N_d$. In contrast, we use an LLM as the question decomposer and input both the ToC c and the original question q into the question decomposer to generate a set of sub-questions $Q' = \{q_1, q_2, \ldots, q_j\}$.

There are two reasons for introducing the ToC: one is to limit the scope of the sub-questions to the themes of the reference document, and the second is to better align the sub-questions with the different sections of the document.

Then, we use Q' for the first layer of clustering, so the number of sub-questions $N_{Q'}$ corresponds to the number of clusters, and the embeddings of the sub-questions $E_{Q'}$ can serve as the initial cluster centers, resulting in the total computational effort:

$$N_d + N_{Q'} + \frac{N_{Q'}}{2} + \cdots \tag{5}$$

Since $N_{Q'} \ll N_d$, the computational effort can be considered approximately N_d, thereby reducing the time required for clustering and generating summaries. Moreover, by reducing the number of clusters, there is no need to employ hierarchical clustering to capture the relationships from themes to details among texts (Sarthi et al., 2024), thus further enhancing the efficiency of clustering.

We provide prompts for decomposing questions in Appendix A.

Text Chunk Segmentation. The complexity of the summary tree scales linearly with the length of the document (Sarthi et al., 2024). Therefore, although there are methods available for segmenting text chunks based on semantics (Sawarkar et al., 2024), given that constructing the summary tree itself incurs significant time overhead, if semantic segmentation is used, then for a dataset such as NarrativeQA (Kočiskỳ et al., 2018)—where a document can exceed 100,000 tokens—the time required to build a summary tree would be intolerable. Therefore, we use a fixed segmentation method, and we set the text chunk size to 500 tokens. Any sentence that extends beyond the 500-token boundary is moved in its entirety to the next text chunk to avoid having an incomplete sentence within a single text chunk.

Clustering. To address the issue of generated summaries being potentially irrelevant to the query, we use the number of sub-questions as the number of clusters, with embeddings of sub-questions serving as initial clustering centers. This quesion-oriented clustering approach enhances the relevance of summaries to the questions.

We use a specific encoder to convert text chunks into embeddings, and then, following previous research (Sarthi et al., 2024), we employ Gaussian Mixture Models (GMM) for soft clustering, allowing a text chunk to be assigned to multiple categories. GMM assumes that all data points are generated from a finite number of Gaussian dis-

tributions, each corresponding to a cluster. The probability density function of the entire model is given by:

$$p(x|\theta) = \sum_{m=1}^{M} \pi_m \mathcal{N}(x|\mu_m, \Sigma_m), \quad (6)$$

where π_m is the mixing weight of the m-th Gaussian distribution, satisfying $0 \leq \pi_m \leq 1$ and $\sum_{m=1}^{M} \pi_m = 1$, and θ represents the set of all parameters to be estimated, including all mixing weights, means, and covariance matrices.

To better measure the similarity between embeddings (Aggarwal et al., 2001), we use Uniform Manifold Approximation and Projection (UMAP) for dimensionality reduction (McInnes et al., 2018). First, we combine all text chunk embeddings $E_T = \{e_{t_1}, e_{t_2}, \ldots, e_{t_p}\}$. and sub-question embeddings $E_{Q'} = \{e_{q_1}, e_{q_2}, \ldots, e_{q_j}\}$ into a new embedding set, then perform unified dimensionality reduction to ensure semantic consistency between sub-questions and text chunks. Formally, this dimensionality reduction process can be expressed as:

$$E_{\text{reduced}} = \Phi_{\text{UMAP}}\left(E_T \oplus E_{Q'}\right) \quad (7)$$

We use global clustering instead of the hierarchical clustering algorithm (Sarthi et al., 2024) because, on one hand, the number of sub-questions is small and hierarchical clustering typically requires a larger number of clusters; on the other hand, global clustering is more efficient. In the first layer of clustering, we use the number of sub-questions as the number of clusters, with embeddings serving as the initial clustering centers. This naturally sets two adaptive hyperparameters for clustering. After the first layer, we do not consider sub-questions but use the number of clusters determined by BIC and random initial clustering centers, as the number of sub-questions by then is greater than or equal to the number of text chunks.

Recursive Summarization Generation. The clustered text chunks are fed into an LLM-based summarization generator to produce respective summaries. This process is then repeated by reducing dimensions and clustering again until further clustering is no longer possible. Although the generated summaries may contain slight hallucinations, these hallucinations do not significantly impact the question-and-answer results (Sarthi et al., 2024; Zhang et al., 2022). The prompts used for generating summaries are provided in Appendix A.

3.3 Retrieval

For the two retrieval methods of the summary tree: tree traversal and collapsed tree (Sarthi et al., 2024), the tree traversal method selects the top k most relevant nodes based on cosine similarity at each layer, considers the children of these selected nodes in the next layer, and then chooses the k nodes with the highest cosine similarity to the query vector, repeating this process until reaching the leaf nodes. The collapsed tree method unfolds all nodes as a single layer, selecting the top k nodes with the highest cosine similarity score to the query, and continuously adds nodes to the result set until a predefined maximum token count is reached. Since the performance of the collapsed tree consistently outperforms tree traversal (Sarthi et al., 2024), we use the collapsed tree method for retrieval in subsequent experiments.

4 Experimental Setup

4.1 Settings

We use GPT-4 (Achiam et al., 2023), GPT-4o-mini, and DeepSeek-V2-Lite-Chat (7B) (Liu et al., 2024a) as the LLMs, which are among the most advanced closed-source and open-source LLMs currently available. The selected LLMs are used for question classification, ToC generation, sub-question generation, summarization, and question answering. The embedding model is SBERT (Reimers, 2019), with the GMM clustering threshold set to 0.5. The maximum length for summaries is 100 tokens, the maximum length for text chunks is 500 tokens. During retrieval, DPR's top-k is set to 5, the maximum token limit for the collapsed tree is set to 3500, and FAISS is used for nearest neighbor search (Johnson et al., 2019). The experiments are conducted on an NVIDIA A800 80GB PCIe GPU and an Intel(R) Xeon(R) Silver 4314 CPU @ 2.40GHz.

4.2 Datasets

Dataset	Number of Questions	Average Document Characters
QASPER	1451	21,889.87
QuALITY	2128	24,723.06
NarrativeQA	10,558	332,133.92

Table 1: Dataset Overview.

Our experiments utilize three question-answering datasets: NarrativeQA (Kočiskỳ et al., 2018), QASPER (Dasigi et al., 2021), and

Type	Number
Extractive	501 (34.53%)
Abstractive	513 (35.35%)
Boolean	239 (16.47%)
Unanswerable	198 (13.65%)

Table 2: Statistics of Different Types of Questions in QASPER.

QuALITY (Pang et al., 2021). The basic statistics of these datasets are presented in Table1 and Table2. These datasets enable the evaluation of DTCRS's performance across various document lengths and diverse question types.

NarrativeQA: This dataset contains reference documents paired with unique question-answer pairs. During evaluation, the model generates answers based on the provided document and question, which are then compared directly to the unique ground-truth answers using standard metrics such as BLEU-1, BLEU-4, ROUGE-L, and METEOR.

QASPER: This dataset comprises shorter documents with multiple answers per question provided by different annotators. Questions are categorized into answerable/unanswerable, yes/no, abstractive, and extractive types. Evaluation is conducted by calculating the token-level F1 score between the model's prediction and all provided answers, selecting the highest score as the final evaluation metric.

QuALITY: QuALITY includes context paragraphs paired with multiple-choice questions. During evaluation, the model selects the most appropriate answer from provided options based on the context and question. The chosen answers are then compared against the gold standard answers to measure performance through accuracy.

4.3 Comparisons

DPR (Karpukhin et al., 2020). Maps queries and documents into a dense vector space and finds the most relevant documents by computing the similarity between vectors.

RAPTOR (Sarthi et al., 2024). Recursively summarizes text in a hierarchical manner, constructing a tree structure that allows the model to retrieve information from different levels of abstraction during inference. This method captures both local details and the overall structure and themes of documents, thereby better supporting complex reasoning tasks.

State-of-the-Art Methods on Each Task. NarrativeQA: BiDAF (Seo, 2016), BM25 + BERT(Robertson et al., 2009; Devlin, 2018), Recursively Summarizing Books (Wu et al., 2021), Retriever + Reader (Izacard and Grave, 2020). **QuALITY**: Longformer-base (Beltagy et al., 2020), DPR + DeBERTaV3-large (Karpukhin et al., 2020; He et al., 2020), CoLISA + DeBERTaV3-large (Dong et al., 2023). **QASPER**: LongT5 XL (Guo et al., 2021), CoLT5 XL (Ainslie et al., 2023).

5 Experimental Results

5.1 Main Results

Model	F1
DPR + GPT-4	51.3
LongT5 XL	53.1
CoLT5 XL	53.9
RAPTOR + GPT-4	55.7
DTCRS + GPT-4	**58.5**

Table 3: Experimental results on QASPER test set using GPT-4 as the language model. The results of other methods are reported as stated in their original papers. DTCRS outperforms RAPTOR and achieves the best results.

Model	Accuracy		SAT-style Score	
	Test Set	Hard Subset	Test Set	Hard Subset
Longformer-base	30.7	29.3	7.6	5.7
DPR + DeBERTaV3-large	55.4	46.1	40.5	28.1
CoLISA + DeBERTaV3-large	62.3	54.7	49.7	39.6
DPR + GPT-4o-mini	62.8	50.3	52	35.9
RAPTOR + GPT-4o-mini	67.6	57	59	44.9
DTCRS + GPT-4o-mini	**74.7**	**62.9**	**63.7**	**48.2**

Table 4: Experimental results on QuALITY test set using GPT-4o-mini as the language model. The results of other methods are reported as stated in their original papers. DTCRS achieves the best performance on both the entire dataset and the challenging subset.

We compare DTCRS with state-of-the-art methods on each task, and the experimental results are the best values from two runs on the test set. As shown in Tables 3 and 4. DTCRS+GPT-4 achieves the highest F1 score of 58.5% on the QASPER dataset, while DTCRS+GPT-4o-mini attains the highest accuracy of 74.7% and 62.9% on the full and hard subsets of the QuALITY dataset, respectively. This demonstrates that the recursive summaries generated by our method provide useful references for LLMs in answering questions.

Model	ROUGE-L	BLEU-1	BLEU-4	METEOR
BiDAF	6.2	5.7	0.3	3.7
BM25 + BERT	15.5	14.5	1.4	5.0
Recursively Summarizing Books	21.6	22.3	4.2	10.6
DPR + GPT-4o-mini	26.5	21.5	1.3	14.2
RAPTOR + GPT-4o-mini	25.0	21.2	1.1	14.1
Retriever + Reader	**32.0**	**35.3**	**7.5**	11.1
DTCRS + GPT-4o-mini	26.0	21.1	1.3	**14.4**

Table 5: Experimental results on NarrativeQA test set using GPT-4o-mini as the language model. The results of other methods are reported as stated in their original papers. DTCRS achieves the best METEOR score but does not show an overall advantage compared to DPR and RAPTOR using the same language model. We attribute this to the lack of complex questions requiring summarization and reasoning in NarrativeQA.

As shown in Table 5, DTCRS+GPT-4o-mini achieves the highest METEOR score of 14.4% on the NarrativeQA dataset but performs worse than the Retriever+Reader (Izacard and Grave, 2020) method on other metrics. We attribute the varying performance of DTCRS across different tasks to differences in question types and difficulty. QASPER contains a significant proportion of abstractive questions that require synthesizing scattered information, while QuALITY includes a challenging hard subset where DTCRS exhibits a more pronounced advantage. Although the NarrativeQA dataset provides an ultra-long document, its questions are predominantly simple extractive questions, such as "Who is Eve?" and "Where does Ralston recover?". These types of questions do not benefit from DTCRS, which explains why DTCRS performs worse than Retriever+Reader and exhibits similar performance to DPR and RAPTOR on this dataset.

We further compare performance on abstractive and non-abstractive questions. As shown in Figure 3, DTCRS consistently outperforms DPR on both types, with larger gains for abstractive questions. These results highlight the particular advantage of DTCRS on tasks that require multi-hop reasoning and information synthesis.

5.2 Ablation Study

We perform ablation experiments using GPT-4o-mini and Deepseek-v2-Lite-Chat on the QASPER dataset to examine the contribution of each module to performance improvement. Specifically, we consider removing (1) global clustering and replacing it with hierarchical clustering, (2) the question type classifier, and (3) the Table of Contents (ToC) generator. The more complete ablation results will be presented in Appendix B.

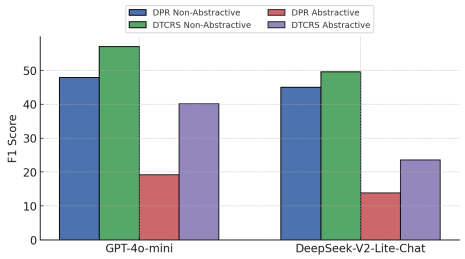

Figure 3: F1 scores comparison of DTCRS and DPR on abstractive vs. non-abstractive questions.

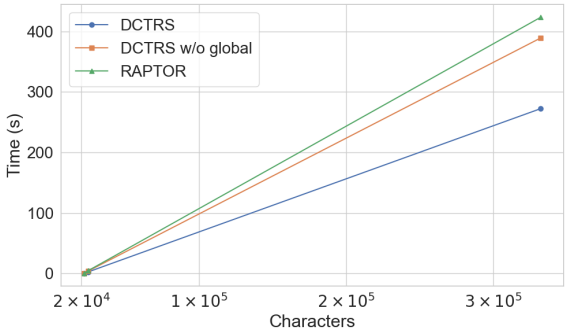

Figure 4: The sample outputs of RAPTOR and DTCRS: the yellow-highlighted parts in the evidence indicate information relevant to the question, the orange-highlighted parts represent redundant evidence included by RAPTOR compared to DTCRS, and the red bold text in the answer denotes the ground truth.

As shown in Table 6, we analyze the performance of different DTCRS modules on various question types and draw the following conclusions: Removing global clustering has no significant impact on the results but improves clustering efficiency. We will analyze the efficiency of global and hierarchical clustering in the next section. Removing the ToC generator leads to a performance drop, indicating that it helps combine sub-questions with query semantics and document structure, improving the relevance between summaries and the question. The most notable impact is from removing the question classifier. This is because for extractive questions, DPR performs better, and introducing recursive summarization in such cases can lead to performance degradation. However, for abstractive questions, recursive summarization is beneficial. Since recursive summarization is not suitable for all types of questions, it is crucial to decide whether to incorporate it based on the question type.

Model	Total		Extractive		Abstractive		Boolean	
	GPT-4o-mini	DeepSeek-V2-Lite-Chat	GPT-4o-mini	DeepSeek-V2-Lite-Chat	GPT-4o-mini	DeepSeek-V2-Lite-Chat	GPT-4o-mini	DeepSeek-V2-Lite-Chat
DPR	33.0	31.3	29.6	25.7	13.0	13.9	86.3	85.4
w/o global	43.8	**38.6**	40.4	32.4	23.5	21.0	86.2	86.2
w/o classify	37.2	33.7	28.0	24.0	22.9	20.2	**90.1**	**87.4**
w/o contents	44.1	38.1	40.5	32.1	22.7	20.6	87.3	84.1
DTCRS	**44.5**	38.3	**41.3**	**32.5**	**23.6**	**21.1**	88.2	85.3

Table 6: Ablation study results on QASPER using GPT-4o-mini and DeepSeek-V2-Lite-Chat, showing the F1 scores of each component on different question types.

Variant	GPT-4o-mini	DeepSeek-V2-Lite-Chat
DPR	33.0	31.3
w/o global	43.8	**38.6**
w/o classifier	37.2	33.7
w/o ToC	44.1	38.1
w/o question decomposer	37.5	32.9
DTCRS	**44.5**	38.3

Table 7: Ablation Study on Total Questions (F1 Scores)

Layer	Average Number of Nodes		Evidence Node Coverage	
	DTCRS	RAPTOR	DTCRS	RAPTOR
0 (leaf)	252	252	75.39%	70.83%
1	3.96	54	19.84%	20.83%
2	1.03	10	3.96%	8.33%

Table 8: Comparison of node statistics across layers. Evidence node coverage is the ratio of the number of evidence nodes in the layer to the total number of evidence nodes.

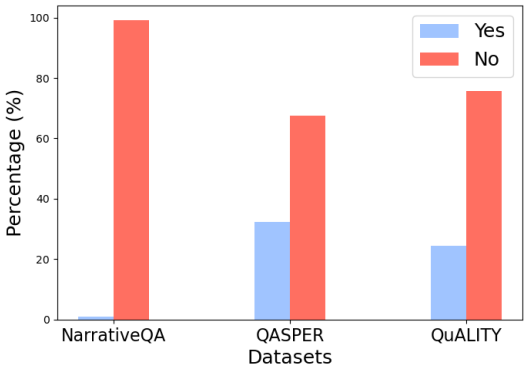

Figure 5: Classification results using GPT-4o-mini as the classifier on different datasets: "Yes" indicates abstractive questions, "No" indicates non-abstractive questions.

5.3 Analysis

To evaluate whether our method reduces the summary tree construction time, we examine the relationship between document length and summary tree construction time, as shown in Figure 4. Overall, all methods exhibit a linear trend. Compared to DTCRS without global clustering and RAPTOR, DTCRS shows increasing time efficiency benefits as document length grows, indicating that our approach effectively reduces summary tree construction time by decreasing the number of clusters and incorporating global clustering.

Furthermore, we analyze the distribution of nodes across different layers to investigate whether DTCRS addresses the issue of redundant summary nodes. As shown in Table 8, we report the average number of nodes and the evidence node coverage rate for the first three layers. DTCRS has significantly fewer summary nodes in the first and second layers than RAPTOR. Meanwhile, the evidence node coverage rate in the leaf layer and the first layer, which play a crucial role in answer generation, remains comparable to RAPTOR. This suggests that DTCRS effectively reduces the number of summary nodes while preserving the query-relevant information, thereby addressing the redundancy issue in summary nodes. The sample outputs in Figure 6 further support this conclusion. Although both DTCRS and RAPTOR can retrieve evidence relevant to the question, DTCRS provides more concise evidence, which helps prevent the generation of irrelevant text in the answer.

In Section 5.1, we find that the experimental results of DTCRS on NarrativeQA are not ideal. We believe this is because NarrativeQA primarily consists of simple extractive questions and lacks abstractive questions. To validate our point, we analyze the classification results of the question classifier on different datasets. As shown in Figure 5, the proportion of abstractive questions in QASPER and QuALITY exceeds 20%, which leads to a significant performance improvement for DTCRS compared to RAPTOR and DPR. The QuALITY dataset benefits even more due to its higher proportion of abstractive questions. In contrast, NarrativeQA has almost no abstractive questions, so the performance of DTCRS is limited.

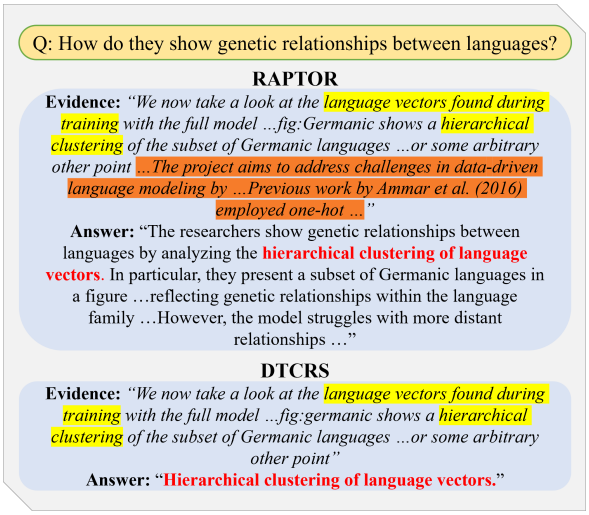

Figure 6: The sample outputs of RAPTOR and DTCRS: the yellow-highlighted parts in the evidence indicate information relevant to the question, the orange-highlighted parts represent redundant evidence included by RAPTOR compared to DTCRS, and the red bold text in the answer denotes the ground truth.

Computational Efficiency. Although our method requires several LLM calls, the main bottleneck is in clustering and summarization. As shown in Table 9, DTCRS reduces summary-layer nodes by 92.2%, and the summary tree construction time (excluding preprocessing) is reduced by 80.95% compared to RAPTOR.

Method	Nodes	Time (s)
DTCRS	4.99	40.82
RAPTOR	64.00	214.39

Table 9: Summary layer node count and construction time (excluding preprocessing) for DTCRS and RAPTOR.

6 Conclusion

We propose DTCRS, which dynamically generates a summary tree based on document structure and query semantics. First, it generates a table of contents for the document and then decomposes complex questions into simpler sub-questions based on the table of contents. The embeddings of these sub-questions serve as the initial cluster centers. Through recursive clustering and text summarization, DTCRS constructs a hierarchical summary tree. DTCRS achieves significant performance improvements on three QA datasets while requiring less time for construction. We conduct experiments to analyze DTCRS's performance on different types of questions, providing insights for future research on the applicability of recursive summarization.

Ethics Statement

We follow the new ACL Policy on AI Writing Assistance and use AI purely for language assistance in the paper. After careful consideration, we believe that our paper complies with the ACL ethics policy and does not introduce any additional ethical concerns.

Limitations

The question classifier may make incorrect classifications, leading to the use of inappropriate retrieval methods, which can degrade performance. For long documents that exceed the model's input limit, the table of contents generated by the ToC generator may be incomplete, limiting the ability to summarize the document information effectively. In addition, due to constraints in computational resources and time, we only conduct experiments using the larger-scale GPT-4 model on the QASPER dataset, while smaller language models are used for the other datasets. Further evaluation with larger models on a broader range of datasets is needed in future work.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful and constructive feedback, which helped us improve the clarity, completeness, and overall quality of our work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. 2001. On the surprising behavior of distance metrics in high dimensional space. In *Database theory—ICDT 2001: 8th international conference London, UK, January 4–6, 2001 proceedings 8*, pages 420–434. Springer.

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontañón, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, et al. 2023. Colt5: Faster long-range transformers with conditional computation. *arXiv preprint arXiv:2303.09752*.

Stefanos Angelidis and Mirella Lapata. 2018. Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised. *arXiv preprint arXiv:1808.08858*.

Manoj Ghuhan Arivazhagan, Lan Liu, Peng Qi, Xinchi Chen, William Yang Wang, and Zhiheng Huang. 2023. Hybrid hierarchical retrieval for open-domain question answering. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10680–10689.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Steven Bird. 2006. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72.

Shuyang Cao and Lu Wang. 2022. Hibrids: Attention with hierarchical biases for structure-aware long document summarization. *arXiv preprint arXiv:2203.10741*.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*.

Zihang Dai. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*.

Yang Deng, Wenxuan Zhang, Weiwen Xu, Ying Shen, and Wai Lam. 2023. Nonfactoid question answering as query-focused summarization with graph-enhanced multihop inference. *IEEE Transactions on Neural Networks and Learning Systems*.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mengxing Dong, Bowei Zou, Yanling Li, and Yu Hong. 2023. Colisa: inner interaction via contrastive learning for multi-choice reading comprehension. In *European Conference on Information Retrieval*, pages 264–278. Springer.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. *arXiv preprint arXiv:2305.14627*.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2021. Longt5: Efficient text-to-text transformer for long sequences. *arXiv preprint arXiv:2112.07916*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Gautier Izacard and Edouard Grave. 2020. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584*.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *arXiv preprint arXiv:2403.14403*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

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

Jiarui Li, Ye Yuan, and Zehua Zhang. 2024. Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases. *arXiv preprint arXiv:2403.10446*.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. 2024a. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

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

Ye Liu, Kazuma Hashimoto, Yingbo Zhou, Semih Yavuz, Caiming Xiong, and Philip S Yu. 2021. Dense hierarchical retrieval for open-domain question answering. *arXiv preprint arXiv:2110.15439*.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Sewon Min, Kenton Lee, Ming-Wei Chang, Kristina Toutanova, and Hannaneh Hajishirzi. 2021. Joint passage ranking for diverse multi-answer retrieval. *arXiv preprint arXiv:2104.08445*.

Fengran Mo, Chen Qu, Kelong Mao, Yihong Wu, Zhan Su, Kaiyu Huang, and Jian-Yun Nie. 2024. Aligning query representation with rewritten query and relevance judgments in conversational search. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1700–1710.

Andrew A Neath and Joseph E Cavanaugh. 2012. The bayesian information criterion: background, derivation, and applications. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(2):199–203.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, et al. 2021. Quality: Question answering with long input texts, yes! *arXiv preprint arXiv:2112.08608*.

Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. 2024. Large language model based long-tail query rewriting in taobao search. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 20–28.

Ansh Radhakrishnan, Karina Nguyen, Anna Chen, Carol Chen, Carson Denison, Danny Hernandez, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, et al. 2023. Question decomposition improves the faithfulness of model-generated reasoning. *arXiv preprint arXiv:2307.11768*.

N Reimers. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Justin Reppert, Ben Rachbach, Charlie George, Luke Stebbing, Jungwon Byun, Maggie Appleton, and Andreas Stuhlmüller. 2023. Iterated decomposition: Improving science q&a by supervising reasoning processes. *arXiv preprint arXiv:2301.01751*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. *arXiv preprint arXiv:2401.18059*.

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. 2024. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. *arXiv preprint arXiv:2404.07220*.

M Seo. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Simeng Sun, Kalpesh Krishna, Andrew Mattarella-Micke, and Mohit Iyyer. 2021. Do long-range language models actually use long-range context? *arXiv preprint arXiv:2109.09115*.

Qingyue Wang, Liang Ding, Yanan Cao, Zhiliang Tian, Shi Wang, Dacheng Tao, and Li Guo. 2023. Recursively summarizing enables long-term dialogue memory in large language models. *arXiv preprint arXiv:2308.15022*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2024. Knowledge editing for large language models: A survey. *ACM Computing Surveys*, 57(3):1–37.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. 2021. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*.

Yumo Xu and Mirella Lapata. 2020. Coarse-to-fine query focused multi-document summarization. In *Proceedings of the 2020 Conference on empirical methods in natural language processing (EMNLP)*, pages 3632–3645.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–184.

Shiyue Zhang, David Wan, and Mohit Bansal. 2022. Extractive is not faithful: An investigation of broad unfaithfulness problems in extractive summarization. *arXiv preprint arXiv:2209.03549*.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*.

Zihan Zhang, Meng Fang, Ling Chen, Mohammad-Reza Namazi-Rad, and Jun Wang. 2023. How do large language models capture the ever-changing world knowledge? a review of recent advances. *arXiv preprint arXiv:2310.07343*.

A Prompt

The prompts used in this study, depicted in Figures 7 to 12, are designed to guide the model through a range of specific tasks effectively.

- **Problem Decomposition:** Figure 7 shows a prompt that decomposes the user's question into sub-questions based on the table of contents.
- **Table of Contents Generation:** Figure 8 prompts the model to create a structured table of contents for the document.
- **Classification of Problem Types:** Figure 9 valuate whether a user's question is complex and requires information from multiple sections of a document to be fully answered.
- **Non-Multiple-Choice Question and Answer:** Figure 10 answers questions concisely (e.g., QASPER, NarrativeQA) using the provided context.
- **Multiple-Choice Question and Answer:** Figure 11 selects the best answer from multiple options based on context (e.g., QuALITY).
- **Summary Generation:** Figure 12 generates a concise summary of the given context, capturing key details.

B Complete Experimental Results

Tables 10 and 11 show the complete experimental results compared with DPR and RAPTOR.

C Sample Output

We present a detailed sample output in Table 12.

Prompt for Problem Decomposition

```
prompt = (
        "Based on the following table of contents, decompose the user's question into multiple "
        "relevant sub-questions that can be answered step by step. The sub-questions should not exceed the scope "
        "of the original question.\n\n"
        f"Table of contents:\n{table_of_contents}\n\n"
        f"User question: {question}\n\n"
        "Please break down the question into sub-questions that correspond to sections in the table of contents. "
        "Do not extend the scope beyond the original question. Format the output as follows:\n"
        "1. Sub-question 1\n"
        "2. Sub-question 2\n"
        "..."
    )
messages=[
            {"role": "system",
             "content": "You are an assistant helping the user decompose questions into sub-questions."},
            {"role": "user", "content": prompt}
          ]
```

Figure 7: Prompt for Problem Decomposition

Prompt for Table of Contents Generation

```
prompt = (
        "Please generate a detailed table of contents for the following document. "
        "The table of contents should include chapter and subchapter titles.\n\n"
        f"Document text: \n{document_text}\n\n"
        "Table of contents:"
    )
messages=[
            {"role": "system",
             "content": "You are an assistant helping the user generate a table of contents."},
            {"role": "user", "content": prompt}
          ]
```

Figure 8: Prompt for Table of Contents Generation

Prompt for Classification of Problem Types

```
prompt = (
        "Given the table of contents listed below, determine if the user's question is complex and "
        "requires information from multiple sections to be fully answered. Answer with 'Yes' if the question is complex, "
        "or 'No' if it is not.\n\n"
        f"Table of contents:\n{table_of_contents}\n\n"
        f"User question: {question}\n"
    )
messages=[
            {"role": "system",
             "content": "You are an assistant tasked with evaluating the complexity of questions."},
            {"role": "user", "content": prompt}
          ]
```

Figure 9: Prompt for Classification of Problem Types

Prompt for Non-Multiple-Choice Question and Answer Tasks

```
prompt = "using the following information {context}. Answer the following question in less than 5-7 words, if possible:
{question}"
messages=[
            {"role": "system", "content": "You are Question Answering Portal"},
            {
              "role": "user",
              "content": prompt,
            },
        ]
```

Figure 10: Prompt for Non-Multiple-Choice Question and Answer Tasks

Prompt for Multiple-Choice Question and Answer Tasks

```
prompt = "Given Context: {context} Choose the best full answer amongst the option to question: {question}"
messages=[
        {"role": "system", "content": "You are Question Answering Portal"},
        {
          "role": "user",
          "content": prompt,
        },
      ]
```

Figure 11: Prompt for Multiple-Choice Question and Answer Tasks

Prompt for Summary Generation

```
messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {
          "role": "user",
          "content": f"Write a summary of the following, including as many key details as possible: {context}:",
        },
      ]
```

Figure 12: Prompt for Summary Generation

Model	Total		Extractive		Abstractive		Boolean	
	GPT-4o-mini	DeepSeek-V2-Lite-Chat	GPT-4o-mini	DeepSeek-V2-Lite-Chat	GPT-4o-mini	DeepSeek-V2-Lite-Chat	GPT-4o-mini	DeepSeek-V2-Lite-Chat
DPR	33.0	31.3	29.6	25.7	13.0	13.9	86.3	85.4
RAPTOR	34.5	34.6	24.4	23.6	17.2	17.7	87.2	86.8
w/o global	43.8	38.6	40.4	32.4	23.5	21.0	86.2	86.2
w/o classify	37.2	33.7	28.0	24.0	22.9	20.2	90.1	87.4
w/o contents	44.1	38.1	40.5	32.1	22.7	20.6	87.3	84.1
DTCRS	**44.5**	**38.3**	**41.3**	**32.5**	**23.6**	**21.1**	**88.2**	**85.3**

Table 10: Complete results on QASPER using GPT-4o-mini and DeepSeek-V2-Lite-Chat.

Model	ROUGE-L		BLEU-1		BLEU-4		METEOR	
	GPT-4o-mini	DeepSeek-V2-Lite-Chat	GPT-4o-mini	DeepSeek-V2-Lite-Chat	GPT-4o-mini	DeepSeek-V2-Lite-Chat	GPT-4o-mini	DeepSeek-V2-Lite-Chat
DPR	26.5	21.3	21.4	15.2	1.3	0.5	14.4	15.0
RAPTOR	25.0	25.4	21.2	17.9	1.1	1.0	14.1	18.3
w/o global	**26.9**	27.9	21.5	18.9	1.3	**1.3**	14.7	19.7
w/o classify	26.2	25.3	21.4	17.3	1.2	0.7	**15.1**	17.7
w/o contents	26.7	21.4	21.4	15.2	1.1	0.6	14.3	15.2
DTCRS	26.0	**28.1**	21.1	**19.1**	**1.3**	0.9	14.2	**20.1**

Table 11: Complete results on NarrativeQA using GPT-4o-mini and DeepSeek-V2-Lite-Chat.

Question	How do they show genetic relationships between languages?	
Reference Answer	By doing hierarchical clustering of word vectors	
Method	**Predicted Answer**	**Predicted Evidence**
RAPTOR	The researchers show genetic relationships between languages by analyzing the hierarchical clustering of language vectors. In particular, they present a subset of Germanic languages in a figure (fig:germanic), where the clustering closely mirrors the established genetic relationships within the Germanic language family. This suggests that the language vectors, while not explicitly designed to detect genealogical relationships, capture meaningful linguistic similarities, reflecting genetic relationships within the language family. However, the model struggles with more distant relationships, such as connecting Germanic languages to Celtic languages, indicating that the model is more successful at capturing closer, more evident linguistic relationships. Therefore, the genetic relationships between languages are demonstrated through clustering, where the model's learned language vectors show a strong correlation with known language family trees, like those of the Germanic languages.	However, our purpose is not to detect genealogical relationships... The only supervision... is a language identifier as a one-hot encoding **In this paper, we examine a language model that utilizes a predictive distribution... Our focus is on investigating the implications of moving beyond the training corpus's language points** By means of cross-entropy... Figure FIGREF12 plots the interpolation results... **The project aims to address challenges in data-driven language modeling... Previous work by Ammar et al. (2016) employed one-hot** The research focuses on developing... **distinguishing it from previous studies, such as Johnson et al. (2016)**...
DTCRS	Hierarchical clustering of language vectors.	However, our purpose is not to detect genealogical relationships... The only supervision... is a language identifier as a one-hot encoding **By means of cross-entropy... Figure FIGREF12 plots the interpolation results...** In the figure English is (weakly) grouped with the West Germanic languages... Generating Text: Since our language model is conditioned on a language vector...

Table 12: Comparison of RAPTOR and DTCRS Results.

D Inference Parameters

Table 13 shows the temperatures for all steps that require reasoning with an LLM.

Process Step	Temperature
Question Type Classification	0
Table of Contents Generation	0
Question Decomposition	0
Question Answering	0
Summary Generation	0.3

Table 13: Temperature settings for different process steps.

E UMAP Parameter Settings and Effects

Key UMAP parameters include `n_neighbors`, `n_components`, and `metric`, which control the number of neighbors for manifold learning, the target dimension for reduction, and the distance metric, respectively. In our main experiments, we set `n_neighbors` to 10, `metric` to cosine, and determine `n_components` as:

$$\texttt{n\_components} = \min(\texttt{dim}, \texttt{len(embeddings)} - 2)$$

where `dim` is set to 10, and `len(embeddings)` denotes the number of text chunk embeddings.

We further evaluated the impact of different UMAP parameters on QASPER, as shown below:

n_neighbors	dim	Predicted F1 Score
10	10	58.5
50	10	56.3
50	50	60.0
100	100	60.8

Table 14: Effects of different UMAP parameters on QASPER performance.

F Evaluation Script

We use the NLTK library (Bird, 2006) to compute ROUGE-L, BLEU, and METEOR scores, where the evaluation parameters for the ROUGE-L score are shown in Table 15. For questions with multiple reference answers, we select the one with the highest score.

Parameter	Value
max_n	4
limit_length	True
length_limit	100
length_limit_type	words
apply_avg	True
apply_best	True
alpha	0.5
weight_factor	1.2
stemming	True

Table 15: Parameter settings for ROUGE-L.