HiChunk: Evaluating and Enhancing Retrieval Augmented Generation with Hierarchical Chunking

Wensheng Lu[1†], **Keyu Chen**[1†], **Zhifeng Shen**[1], **Ruizhi Qiao**[1*], **Xing Sun**[1]
[1]Tencent Youtu Lab
{vincentwslu,yolochen,billsshen,ruizhiqiao,winfredsun}@tencent.com

Abstract

Retrieval-Augmented Generation (RAG) enhances the response capabilities of language models by integrating external knowledge sources. However, document chunking as an important part of RAG system often lacks effective evaluation tools. This paper first analyzes why existing RAG evaluation benchmarks are inadequate for assessing document chunking quality, specifically due to evidence sparsity. Based on this conclusion, we propose HiCBench, which includes manually annotated multi-level document chunking points, synthesized evidence-dense question answer(QA) pairs, and their corresponding evidence sources. We also propose HiChunk, a hierarchical document structuring framework using fine-tuned LLMs and the Auto-Merge retrieval algorithm to enhance retrieval quality. Experiments demonstrate that HiCBench effectively evaluates the impact of different chunking methods across the entire RAG pipeline. Moreover, HiChunk achieves better chunking quality within reasonable time consumption, thereby enhancing the overall performance of RAG systems. Source code is available at `https://github.com/TencentCloudADP/hichunk`.

1 Introduction

RAG (Retrieval-Augmented Generation) enhances the quality of LLM responses to questions beyond their training corpus by flexibly integrating external knowledge through the retrieval of relevant content chunks as prompts(Lewis et al., 2020). This approach helps reduce hallucinations(Chen et al., 2024b; Zhang et al., 2025), especially when dealing with real-time information(He et al., 2022) and specialized domain knowledge(Wang et al., 2023; Li et al., 2023). Document chunking, a crucial component of RAG systems, significantly impacts the quality of retrieved knowledge and, consequently, the quality of responses. Poor chunking methods may separate continuous fragments, leading to information loss, or combine unrelated information, making it more challenging to retrieve relevant content. For instance, as noted in Bhat et al. (2025), the optimal chunk size varies significantly across different datasets.

Although numerous benchmarks exist for evaluating RAG systems(Bai et al., 2024; Dasigi et al., 2021; Duarte et al., 2024; Zhang et al., 2024; Yang et al., 2018b; Kočiskỳ et al., 2018; Pang et al., 2021), they mostly focus on assessing either the retriever's capability or the reasoning ability of the response model, without valid evaluation of chunking methods. We analyzed several datasets to determine the average word and sentence count of evidence. As shown in Figure 1, existing benchmarks generally suffer from evidence sparsity, where only a few sentences in the document are relevant to the query. As illustrated in Figure 1, this sparsity of evidence makes these datasets inadequate for evaluating the performance of chunking methods. In reality, user tasks might be evidence-dense, such as enumeration or summarization tasks, requiring chunking methods to accurately and completely segment semantically continuous fragments. Therefore, it is essential to effectively evaluate chunking methods.

To address this, we introduce **Hi**erarchical **C**hunking Benchmark(**HiCBench**), a benchmark for document QA designed to evaluate the impact of chunking methods on different components of RAG systems, including the performance of document chunking, retrievers, and response models. HiCBench's original documents are sourced from OHRBench. We curated documents of appropriate length for the corpus and manually annotated chunking points at various hierarchical levels for evaluation purposes. These points are used to assess the chunker's performance and construct QA pairs, followed by using LLMs and the annotated

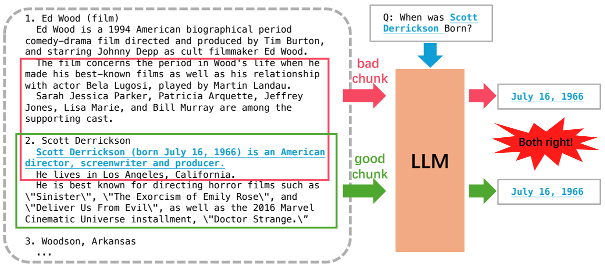
---

[†] These authors contributed equally to this work
[*] Correspondence

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 29738–29753
July 2-7, 2026

Table 1: Statistics of benchmarks.

Dataset	Qasper	OHRBench	GutenQA
Num_{doc}	416	1261	100
Sent_d	164	176	5,373
Word_d	4.2k	5.4k	146.5k
Num_{qa}	1,372	8,498	3,000
Word_q	8.9	20.6	16.0
Word_a	16.0	5.6	26.0
Word_e	239.4	36.5	39.3
Sent_e	10.5	1.7	1.7



1. Ed Wood (film)
Ed Wood is a 1994 American biographical period comedy-drama film directed and produced by Tim Burton, and starring Johnny Depp as cult filmmaker Ed Wood.
The film concerns the period in Wood's life when he made his best-known films as well as his relationship with actor Bela Lugosi, played by Martin Landau.
Sarah Jessica Parker, Patricia Arquette, Jeffrey Jones, Lisa Marie, and Bill Murray are among the supporting cast.

2. Scott Derrickson
Scott Derrickson (born July 16, 1966) is an American director, screenwriter and producer.
He lives in Los Angeles, California.
He is best known for directing horror films such as \"Sinister\", \"The Exorcism of Emily Rose\", and \"Deliver Us From Evil\", as well as the 2016 Marvel Cinematic Universe installment, \"Doctor Strange.\"

3. Woodson, Arkansas
...

Q: When was Scott Derrickson Born?

Figure 1: Different methods produce the same answer.

document structure to create evidence-dense QA, and finally extracting relevant evidence sentences and filtering non-compliant samples using LLMs.

Additionally, existing document chunking methods only consider linear document structure(Duarte et al., 2024; Xiao et al., 2024; Zhao et al., 2025; Wang et al., 2025), while user problems may involve fragments with different semantic granularity, and linear document structure makes it difficult to adaptively adjust during retrieval. Therefore, we propose the **Hi**erarchical **C**hunking framework(**HiChunk**), which employs fine-tuned LLMs for hierarchical document structuring and incorporates iterative reasoning to address the challenge of adapting to extremely long documents. For hierarchically structured documents, we introduce the Auto-Merge retrieval algorithm, which adaptively adjusts the granularity of retrieval chunks based on the query, thereby maximizing retrieval quality. In this work, our main contributions are as follows:

- We introduce HiCBench, a benchmark designed to assess the performance of chunker and the impact of chunking methods on retrievers and response models within RAG systems. HiCBench includes information on chunking points at different hierarchical levels of documents, as well as sources of evidence and factual answers related to evidence-dense QA, enabling better evaluation of chunking methods.
- We propose the HiChunk framework, a document hierarchical structuring framework that allows RAG systems to dynamically adjust the semantic granularity of retrieval chunks.
- We conduct comprehensive performance evaluations on several open-source datasets and HiCBench, analyzing the impact of different chunking methods across three dimensions: performance of chunker, retriever, and responder.

2 Related Works

Traditional Text Chunking. Text chunking divides continuous text into meaningful units like sentences, phrases, and words, with our focus on sentence-level chunking. Recent works have explored various approaches: (Cho et al., 2022) combines text chunking with extractive summarization using hierarchical representations and determinantal point processes (DPPs) to minimize redundancy, (Liu et al., 2021) presents a pipeline integrating topical chunking with hierarchical summarization, and (Zhang et al., 2021) develops an adaptive sliding-window model for ASR transcripts using phonetic embeddings. However, these LSTM and BERT(Devlin et al., 2019) based methods face limitations from small context windows and single-level chunking capabilities.

RAG-oriented Document Chunking. Recent research has explored content-aware document chunking strategies for RAG systems. LumberChunker(Duarte et al., 2024) uses LLMs to identify semantic shifts,but may miss hierarchical relationships. PIC(Wang et al., 2025) proposes pseudo-instruction for document chunking, guide chunking via document summaries, though its single-level approach may oversimplify document structure. AutoChunker(Jain et al., 2025) employs tree-based representations but primarily focuses on noise reduction rather than multi-level granularity. Late Chunking(Günther et al., 2024) embeds entire documents before chunking to preserve global context, but produces flat chunk lists without modeling hierarchical relationships. LongRefiner(Jin et al., 2025) introduced two-level chunking, but it is constrained by the model input length and hallucination issues. In contrast, our HiChunk method creates multi-level document representations, chunking from coarse sections to fine-grained paragraphs. This enables RAG systems to retrieve information at appropriate abstraction levels, bridging fragmented knowledge gaps.

Limitations of Existing Text Chunking Benchmarks. The evaluation of text chunking and RAG methods heavily relies on benchmark datasets. Wiki-727k(Koshorek et al., 2018),VT-SSum(Lv et al., 2021) and NewsNet(Wu et al., 2023) are typically chunked into flat sequences of paragraphs or sentences, without capturing the multi-level organization (e.g., sections, subsections, paragraphs) inherent in many real-world documents. This single-level representation limits the ability to evaluate chunking methods that aim to preserve or leverage document hierarchy, which is crucial for comprehensive knowledge retrieval in complex RAG scenarios. While Qasper(Dasigi et al., 2021), HotpotQA(Yang et al., 2018a) and GutenQA(Duarte et al., 2024) are designed for RAG-related tasks, they do not specifically provide mechanisms or metrics for evaluating the efficacy of document chunking strategies themselves. Their focus is primarily on end-to-end RAG performance, where the impact of chunking is implicitly measured through retrieval and generation quality. This makes it challenging to isolate and assess the performance of different chunking methods independently, hindering systematic advancements in hierarchical document chunking. Our work addresses these gaps by proposing a method that explicitly considers multi-level document chunking and constructs a novel benchmark from a chunking perspective.

3 HiCBench Construction

In order to construct the HiCBench dataset, we performed additional document hierarchical structuring and created QA pairs to evaluate document chunking quality, building on the OHRBench document corpus(Zhang et al., 2024). It contains documents from various fields in the real world, such as academia, finance, law, manual, and so on. We filter documents with fewer than 4,000 words and those exceeding 50 pages. For retained documents, we manually annotated the hierarchical structure and used these annotations to assist in the generation of QA pairs and to assess the accuracy of document chunking.

Task Criteria To ensure that the constructed QA pairs could effectively evaluate the quality of document chunking, we aimed for the evidence associated with each QA pair to be widely distributed across a complete semantic chunk. Failure to fully recall such a semantic chunk would result in missing evidence, thereby degrading the quality of the generated responses. To achieve this objective, we established the following standards to regulate the generation of QA pairs:

- **Evidence Completeness and Density**: Evidence completeness ensures that the evidence relevant to the question is comprehensive and necessary within the context. Evidence density requires that evidence constitutes a larger proportion of the context, enhancing the QA pair's utility for evaluating chunking methods.
- **Fact Consistency**: To ensure the constructed samples can evaluate the entire retrieval-based pipeline, it is essential that the generated responses remain consistent with the answers when provided with full context, and that the questions are answerable.

Task Definition We define three different task types to evaluate the quality of chunking:

- **Evidence-Sparse QA** (T_0): The evidence related to the QA is confined to one or two sentences within the document.
- **Single-Chunk Evidence-Dense QA** (T_1): Evidence sentences related to the QA constitute a substantial portion of the context within a single complete semantic chunk. The chunk size ranges from 512 to 4096 tokens.
- **Multi-Chunk Evidence-Dense QA** (T_2): Evidence sentences related to the QA are distributed across multiple complete semantic chunks, covering a significant portion of the context. The chunk size ranges from 256 to 2048 tokens.

QA Construction We use a prompt-based approach using DeepSeek-R1-0528[*] to generate candidate QA pairs, followed by a series of filtering processes to ensure the retained QA pairs meet the criteria of evidence completeness, density, and fact consistency. The specific process is as follows:

1. **Document Hierarchical Annotation and Summarization**: To enable LLMs to gain an overall understanding of the specific document D while constructing QA pairs, we first generated summaries for corresponding sections based on the annotated hierarchical structure, denoted as $S \leftarrow LLM_s(D)$. These summaries will be used in QA pair generation.

---

[*] https://huggingface.co/deepseek-ai/DeepSeek-R1-0528

2. **Generation of Questions and Answers**: We randomly selected one or two chunks from all eligible document fragments as context C, then generated candidate QA pairs using (S, C), where $(Q, A) \leftarrow LLM_{qa}(S, C)$.

3. **Ensuring Evidence Completeness and Density**: Referring to Friel et al. (2024), we use LLMs to extracted sentences from context C related to the QA pair as evidence, denoted as $E \leftarrow LLM_{er}(C, Q, A)$. To mitigate hallucination effects, this step will be repeated five times, retaining sentences that appeared at least four times as the final evidence. Furthermore, to ensure evidence density, we remove samples which the ratio of evidence is less than 10% of context C.

4. **Ensuring Fact Consistency**: We applied Fact-Cov metric(Xiang et al., 2025) to filter test samples. We first extract the facts from answer A, denoted as $F \leftarrow LLM_{fe}(Q, A)^*$. Contexts C used for constructing QA pairs will be provided to LLMs to generate response R', denoted as $R' \leftarrow LLM_r(Q, C)$. Then, the Fact-Cov metric will be calculated by Fact_Cov $\leftarrow LLM_{fc}(F, R')^*$. This process will be repeated 5 times. We retain samples with an average Fact-Cov metric exceeding 80%. Samples below this threshold are deemed unanswerable. All prompts used for QA construction are provided in subsection A.7.

4 HiChunk Framework

This section primarily introduces the HiChunk framework. The overall framework is illustrated in Figure 2. The aim is for the fine-tuned LLMs to comprehend the hierarchical relationships within a document and ultimately organize the document into a hierarchical structure. This involves two subtasks: identification of chunking points and determination of hierarchy levels. Through prompts, HiChunk converts these two subtasks into text generation task. In model train of HiChunk, we use Gov-report(Huang et al., 2021), Qasper(Dasigi et al., 2021) and Wiki-727k(Koshorek et al., 2018) to construct training instructions, which are publicly available datasets with explicit document structure. Meanwhile, we augment the training set by randomly shuffling document chapters and deleting document content.

During inference, HiChunk first splits a document D into a list of sentences $S = [s_1, s_2, ..., s_N]$ (each sentence is assigned a unique ID). The goal is to output a set of hierarchical chunk points that partition S into non-overlapping, semantically complete chunks. Each chunk point is represented as a tuple: $(id, level)$, it represents a semantic break at a specific hierarchy level.

Although the chunking result of HiChunk has semantic integrity, the variability in the chunk length distribution caused by the semantic chunking method can lead to disparities in semantic granularity, which can affect retrieval quality. To mitigate this, we apply a fixed-size chunking approach on the results of HiChunk to produce $C_{[1:M]}$, and propose the Auto-Merge retrieval algorithm to balance issues of varying semantic granularity and the semantic integrity of retrieved chunks.

Iterative Inference For documents exceeding the model's input length limit L, we employ a sliding window approach. In each iteration, we greedily select the longest possible text segment starting from the current position that fits within the limit L. The model then predicts local chunk points for this segment, which are subsequently aggregated into the global document structure.

However, iterative inference suffers from hierarchical drift phenomenon. Due to the lack of complete structural information about document, the model may incorrectly predict the first chunking point of the current inference process as a level-1 segment, thereby causing local hierarchical misalignment. To mitigate this problem, we construct residual text lines from known document structures to guide the model making correct hierarchical judgments. Figure 3 represents a simple case to illustrate how the residual lines work. In short, HiChunk compress the predicted document structure, and bring it into subsequent inference to maintain the model's awareness of the global document structure. The complete iterative inference procedure is illustrated in algorithm 1.

Auto-Merge Retrieval Algorithm To balance the semantic richness and completeness of recalled contexts, we propose Auto-Merge retrieval algorithm. This algorithm uses a series of conditions to control the extent to which child nodes are merged upward into parent nodes. Auto-Merge algorithm traverses the query-ranked chunks $C^{sorted}_{[1:M]}$, using

---

[*] https://github.com/GraphRAG-Bench/GraphRAG-Benchmark

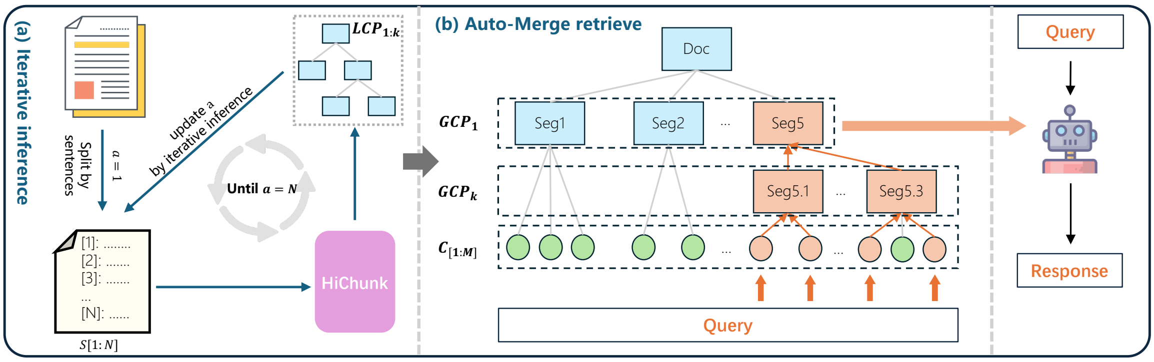


Figure 2: Overview of the proposed HiChunk framework. (a) Iterative inference pipeline: The document is split into sentences; we iteratively predict hierarchical chunk points within the model's context limit. (b) Auto-Merge retrieval: Based on the hierarchical chunks, the algorithm adaptively merges fine-grained child nodes into coarse-grained parent nodes to balance semantic integrity and retrieval granularity.

$\mathcal{N}$ to record the nodes that have been recalled. During the i-th step of the traversal, we first record the current used token budget, $T_{used} = \sum_{n\in\mathcal{N}} \text{len}(n)$. We then add $C_{[i]}^{sorted}$ to $\mathcal{N}$ and denote the parent of $C_{[i]}^{sorted}$ by p. Finally, we merge upward when the following conditions are met:

- **Coherence ($Cond_1$):** The retrieval set contains multiple children from the same parent. Formally, the number of retrieved children must be at least two: $|\mathcal{N} \cap \text{children}(p)| \geq 2$.
- **Substantiality ($Cond_2$):** The total length of the retrieved children covers a significant portion of the parent text. We require $\sum_{n\in(\mathcal{N}\cap\text{children}(p))} \text{len}(n) \geq \theta^* * \text{len}(p)$. Here, θ^* is an adaptive threshold defined as:

$$\theta^*(T_{used}, p) = \frac{1}{3} \times \left(1 + \frac{T_{used}}{T_{max}}\right)$$

  where T_{used} is the token used and T_{max} is the total budget. This design ensures that θ^* starts low and increases as the budget fills up. Intuitively, this encourages **higher-ranking chunks** (when T_{used} is low) to merge more aggressively, prioritizing structural integrity for the more relevant chunks.
- **Feasibility ($Cond_3$):** The remaining token budget is sufficient to accommodate the full parent node after replacing its children.

The entire retrieval algorithm process is illustrated in algorithm 2.

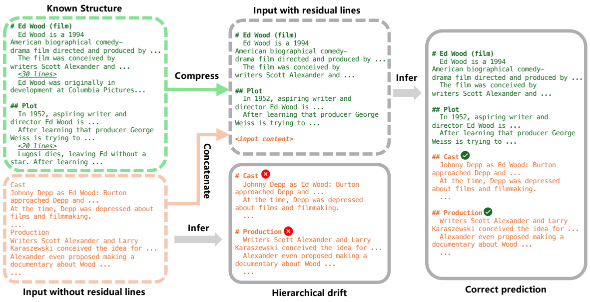


Figure 3: Illustration of how residual lines work in iterative inference.

5 Experiments

5.1 Datasets and Metrics

The test subsets of Gov-report(Huang et al., 2021) and Qasper(Dasigi et al., 2021) datasets will be used for evaluation of chunking accuracy. For the Gov-report dataset, we only retain documents with document word count greater than 5k for experiments. To evaluate the accuracy of the chunking points, we use the $F1$ metrics of the chunking points. The $F1_{L_1}$ and $F1_{L_2}$ correspond to the chunking points of the level 1 and level 2 chunks, respectively. And the $F1_{L_{all}}$ metric does not consider the level of the chunking point. The Qasper, GutenQA(Duarte et al., 2024), and OHRBench(Zhang et al., 2024) datasets contain evidence relevant to the question. These datasets will be used in the evaluation for context retrieval.

For the full RAG pipeline evaluation, we used the publicly available datasets LongBench(Bai et al., 2024), Qasper, GutenQA, and OHRBench. We use the F1 score and Rouge metrics to assess the quality of LLM responses. All experiments are conducted in the code repository of LongBench[*].

Furthermore, HiCBench will be used for comprehensive evaluation, including chunking accuracy, evidence recall rate, and RAG response quality assessment. To avoid biases from sparse text quality evaluation metrics, we employ the Fact-Cov(Xiang et al., 2025) metric for response quality evaluation of HiCBench. The Fact-Cov metric is repeatedly calculated 5 times to take the average. Statistics information of datasets used are shown in Table A1.

5.2 Comparison Methods

We primarily compared two types of chunking methods: rule-based chunking methods and semantic-based chunking methods. All the comparison methods are as follows:

- **FC200**: Fixed chunking is a rule-based method, which first divide the document into sentences and then merge sentences based on a fixed chunking size. Here, the fixed chunking size is 200.
- **SC**: Semantic Chunker(Xiao et al., 2024) uses an embedding model to calculate the similarity between adjacent paragraphs for chunking. We use bge-large-en-v1.5(Xiao et al., 2024) as the embedding model.
- **LC**: LumberChunker(Duarte et al., 2024) employs LLMs to predict the positions for chunking. In our experiments, we use Deepseek-r1-0528(DeepSeek-AI, 2025) as the prediction model. The sampling temperature set to 0.1.
- **HC200**: HiChunk is the proposed method. In the model training for HiChunk. We further chunk the chunks of HiChunk by the fixed chunking method. The fixed chunking size is set to 200, denoted as HC200.
- **HC200+AM**: "+AM" represents the result of introducing Auto-Merge retrieval algorithm on the basis of HC200.

5.3 Experimental Settings

In the model training of HiChunk, Gov-report(Huang et al., 2021), Qasper(Dasigi et al., 2021) and Wiki-727k(Koshorek et al., 2018) are the train datasets, which are publicly available datasets with explicit document structure. We use Qwen3-4B(Team, 2025) as the base model, with a learning rate of 1e-5 and a batch size of 64. The maximum length of training and inference is set to 8192 and 16384 tokens, respectively. Meanwhile, the length of each sentence is limited to within 100 characters. Due to the varying sizes of chunks resulting from semantic-based chunking, we limit the length of the retrieved context based on the number of tokens rather than the number of chunks for a fair comparison. The maximum length of the retrieved context is set to 4096 tokens. We also compare the performance of different chunking methods under different retrieved context length settings in subsection 5.6. In the RAG evaluation process, we consistently use Bge-m3(Chen et al., 2024a) as the embedding model for context retrieval. As for the response model, we use three different series of LLMs with varying scales: Llama3.1-8B(Dubey et al., 2024), Qwen3-8B, and Qwen3-32B.

Table 2: Chunking accuracy. **HC** means HiChunk without fixed-size chunking. The best result is in **bold**.

Chunk Method		**SC**	**LC**	**HC**
Qasper	$F1_{L_1}$	0.0759	0.5481	**0.6742**
	$F1_{L_2}$	-	-	**0.5169**
	$F1_{L_{all}}$	0.1007	0.6657	**0.9441**
Gov-Report	$F1_{L_1}$	0.0298	0.1795	**0.9505**
	$F1_{L_2}$	-	-	**0.8895**
	$F1_{L_{all}}$	0.0616	0.5631	**0.9882**
HiCBench	$F1_{L_1}$	0.0487	0.2849	**0.4841**
	$F1_{L_2}$	-	-	**0.3140**
	$F1_{L_{all}}$	0.1507	0.4858	**0.5450**

5.4 Chunking Accuracy

To comprehensively evaluate the performance of the semantic-based chunking method, we conducted experiments using two publicly available datasets, along with the proposed benchmark, to assess the cut-point accuracy of the chunking method. Since the SC and LC chunking methods are limited to performing single-level chunking, we evaluated only the F1 scores for the initial level of chunking points and the F1 scores without regard for the hierarchy of chunking points. The evaluation results are presented in Table 2. In the Qasper and Gov-report datasets, which serve as in-domain test sets, the HC method shows a significant improvement in chunk accuracy compared to the SC and LC methods. Additionally, in HiCBench, an out-of-domain test set, the HC method exhibits even more substantial accuracy improvements. These findings demonstrate that HC enhances the base model's performance in document chunking by fo-

---

[*] https://github.com/THUDM/LongBench/tree/main

Table 3: RAG-pipeline evaluation results (ERec: Evidence Recall, FC: Fact Coverage). The best result is in **bold**, and the sub-optimal result is in underlined

Chunk	**LongBench**	**Qasper**		**GutenQA**		**OHRBench(T_0)**		**HiCBench(T_1)**			**HiCBench(T_2)**		
Method	Score	ERec	F1	ERec	Rouge	ERec	Rouge	ERec	FC	Rouge	ERec	FC	Rouge
						Llama3.1-8B							
FC200	42.49	84.08	47.26	64.43	30.03	67.03	51.01	74.84	47.82	28.43	74.61	46.79	30.97
SC	42.12	82.08	47.47	58.30	28.58	62.65	49.10	72.14	46.80	28.43	73.49	45.28	30.92
LC	42.73	87.08	48.20	63.67	30.22	**68.42**	51.85	76.64	50.84	29.62	76.12	49.12	32.01
HC200	**43.17**	86.16	48.09	65.13	29.95	68.25	51.33	78.52	49.87	29.38	78.76	49.11	31.80
+AM	42.90	**87.49**	**48.95**	**65.47**	**30.33**	67.84	**51.92**	**81.59**	**55.58**	**30.04**	**80.96**	**53.66**	**33.04**
						Qwen3-8B							
FC200	43.95	84.32	45.10	64.50	33.47	67.07	48.18	74.06	47.35	33.83	72.95	43.45	35.27
SC	43.54	82.22	44.55	58.37	32.71	62.18	46.79	71.42	46.07	33.30	72.36	42.97	34.76
LC	**44.83**	87.43	**46.05**	63.67	33.87	**68.79**	49.28	75.53	48.27	34.12	75.14	46.80	35.93
HC200	43.90	86.49	45.95	65.20	33.89	68.57	49.06	77.68	47.37	34.30	78.10	46.20	36.32
+AM	44.41	**87.85**	45.82	**65.53**	**34.15**	68.31	**49.61**	**81.03**	**50.75**	**35.26**	**80.65**	**49.02**	**37.28**
						Qwen3-32B							
FC200	46.33	84.32	46.49	64.50	44.86	67.07	46.89	74.06	63.20	35.70	72.95	60.87	37.17
SC	46.29	82.22	46.39	58.37	43.59	62.18	45.43	71.26	61.09	35.64	72.36	59.23	37.09
LC	**47.43**	87.43	46.82	63.67	44.45	**68.79**	**47.92**	75.53	64.76	36.15	75.14	62.75	38.02
HC200	46.71	86.49	46.99	65.20	44.83	68.57	47.71	77.68	63.93	36.55	78.10	62.51	38.26
+AM	46.92	**87.85**	**47.25**	**65.53**	**44.94**	68.31	47.89	**81.03**	**68.12**	**37.29**	**80.65**	**66.36**	**39.37**

cusing exclusively on the chunking task. Moreover, as indicated in the subsequent experimental results presented in subsection 5.5, the accuracy improvement of the HC method in document chunking leads to enhanced performance throughout the RAG pipeline. This includes improvements in the quality of evidence retrieval and model responses.

5.5 RAG-pipeline Evaluation

We evaluated the performance of various chunking methods on the LongBench, Qasper, GutenQA, OHRBench, and HiCBench datasets, with the results detailed in Table 3. The results demonstrate that the HC200+AM method achieves either optimal or suboptimal performance on most LongBench subsets. When considering average scores, LumberChunk remains a strong baseline. However, as noted in Table A1, both GutenQA and OHRBench datasets exhibit the feature of evidence sparsity, meaning that the evidence related to QA pairs is derived from only a few sentences within the document. Consequently, the different chunking methods show minimal variation in evidence recall and response quality metrics on these datasets. For instance, using Qwen3-32B as the response model on the GutenQA dataset, the evidence recall metrics of FC200 and HC200+AM are 64.5 and 65.53, and the Rouge metrics are 44.86 and 44.94, respectively. Another example is OHRBench dataset, the evidence recall metrics and Rouge metrics of FC200, LC, HC200 and HC200+AM are very close. In contrast, the Qasper and HiCBench datasets contain denser evidence, where a better chunking method results in higher evidence recall and improved response quality. Again using Qwen3-32B as an example, on the T_1 task of HiCBench dataset, the evidence recall metric for FC200 and HC200+AM are 74.06 and 81.03, the Fact-Cov metrics are 63.20 and 68.12, and the Rouge metrics are 35.70 and 37.29, respectively. These findings suggest that the evidence-dense QA in the HiCBench dataset is better suited for evaluating the quality of chunking methods, enabling researchers to identify bottlenecks within the overall RAG pipeline.

5.6 Influence of Retrieval Token Budget

Since HiCBench is more effective in assessing the performance of chunking methods, we evaluated the impact of our proposed method on the T_1 task of HiCBench under different retrieve token budgets: 2k, 2.5k, 3k, 3.5k and 4k tokens. We compared the effects of various chunking methods by calculating the Rouge metrics between responses and answers, as well as the Fact-Cov metrics. The experimental findings are illustrated in Figure 4. The results demonstrate that a larger retrieval token budget usually leads to better response quality, so it is necessary to compare different chunking methods under the same retrieval token budget. HC200+AM consistently achieves superior response quality across various retrieve token budget settings. These experimental results underscore the effectiveness of HC200+AM method. We further present the correspond curves of the evidence recall metrics in subsection A.4.

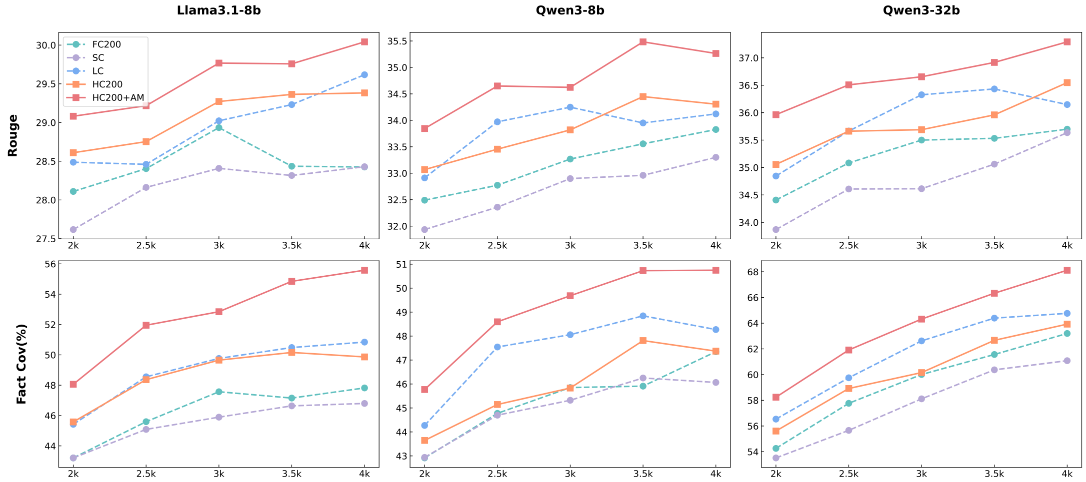


Figure 4: Performance of HiCBench(T_1) under different retrieval token budget from 2k to 4k.

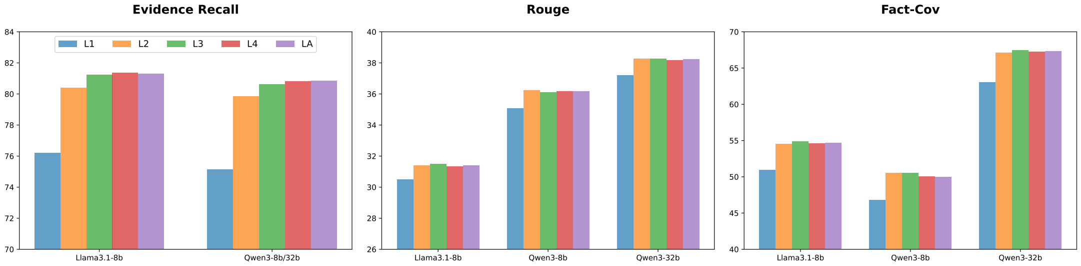


Figure 5: Evidence recall metric across different maximum level on HiCBench(T_1 and T_2).

5.7 Effect of Maximum Hierarchical Level

In this section, we examine the impact of limiting the maximum hierarchical level of document structure obtained by HiChunk. The maximum level ranges from 1 to 4, denoted as $L1$ to $L4$, while LA represents no limitation on the maximum level. We measure the evidence recall metric on different settings. As shown in Figure 5. This result reveals that the Auto-Merge degrades the performance of RAG system in the $L1$ setting due to the overly coarse-grained semantics of $L1$ chunks. As the maximum level increases from 1 to 3, the evidence recall metric also gradually improves and remains largely unchanged thereafter. These findings highlight the importance of document hierarchical structure for enhancing RAG systems.

5.8 Ablation Study for Auto-Merge

To verify the necessity and robustness of the rule design in the Auto-Merge algorithm, we conducted ablation experiments on its core merging conditions, using Qwen3-8B as the generator. The results are presented in Table 4.

Table 4: Ablation study for merging conditions of Auto-Merge.

Dataset	Metrics	Condition Combination $Cond_3$ **Only**	 $Cond_{1+3}$	 $Cond_{1+2+3}$
HiCBench	ERec	81.43	80.55	80.86
	Rouge	36.33	36.08	36.17
	Fact-Cov	51.35	50.70	49.97
LongBench	Score	43.25	43.80	44.41
Qasper	ERec	86.73	87.54	87.85
	F1	45.29	45.83	45.82
OHRBench	ERec	66.72	68.18	68.31
	Rouge	48.78	49.56	49.61

When only $Cond_3$ (token budget constraint) is retained, the algorithm achieves optimal performance on evidence-dense tasks (HiCBench), with ERec of 81.43, Rouge of 36.33, and Fact-Cov of 51.35. However, its performance degrades notably on evidence-sparse tasks: the LongBench Score drops to 43.25, and the ERec on OHRBench is only 66.72. This indicates that relying solely on a single rule leads to poor generalization across diverse task types, lacking sufficient robustness.

After adding $Cond_1$ (semantic intersection constraint), the ERec of Qasper and OHRBench increases by 0.81 and 1.46, respectively, proving that semantic intersection constraints can mitigate "meaningless merging", thereby enhancing retrieval accuracy for evidence-sparse tasks.

With the addition of $Cond_2$ (length ratio constraint), the performance across all datasets tends to be balanced: LongBench Score increases to 44.41 (increases by 1.16), while HiCBench performance only slightly decreases (ERec decrease by 0.57). These results confirm that the combination of multiple complementary rules enables the Auto-Merge algorithm to adapt to both evidence-dense and evidence-sparse tasks, significantly improving its robustness. Furthermore, we conducted a sensitivity analysis on the threshold θ^* of $Cond_2$, and the detailed results are provided in subsection A.3.

5.9 Auto-Merge on Flat vs. Hierarchical Chunking

We conduct an ablation study to disentangle whether the performance gain originates from the hierarchical chunk structure or the Auto-Merge (AM) heuristic alone. We apply AM to flat single-level chunking methods (SC, LC) and the first-level-only variant of our approach (HC$_{L1}$), and compare them with our full hierarchical chunking (HC) on HiCBench, using Llama3.1-8B as the generator.

Table 5: Ablation: Auto-Merge on flat vs. hierarchical chunking (Llama3.1-8B on HiCBench). $_\triangle$ denotes performance change after applying Auto-Merge.

Method	**ERec$_\triangle$**	**Rouge$_\triangle$**	**Fact-Cov$_\triangle$**
SC+AM	$73.35_{+0.60}$	$29.48_{-0.07}$	$45.92_{-0.19}$
LC+AM	$76.86_{+0.46}$	$30.81_{+0.10}$	$50.21_{+0.16}$
HC$_{L1}$+AM	$77.66_{+0.96}$	$30.45_{+0.05}$	$49.68_{+0.42}$
HC+AM	$\mathbf{81.30}_{+2.67}$	$\mathbf{31.41}_{+0.94}$	$\mathbf{54.70}_{+5.17}$

As shown in Table 5, applying AM to flat single-level chunking (SC, LC, HC$_{L1}$) yields negligible or no improvements across all metrics. This demonstrates that the AM heuristic cannot effectively improve retrieval quality when operating on non-hierarchical, flat chunk outputs. In contrast, our full hierarchical chunking (HC) combined with AM achieves large and consistent gains: ERec improves by 2.67 over HC alone.

These results confirm that the strong performance of our method stems from the synergy between hierarchical document structuring and the Auto-Merge retrieval algorithm, rather than the merging heuristic in isolation.

5.10 Few-shot Prompting Experiments

To verify the necessity of fine-tuning, we have supplemented few-shot prompted experiments on HiCBench dataset. The base model is Qwen3-4B (consistent with the base model in the paper). We set two scenarios (1-shot and 3-shot) to compare their performance with the fine-tuned HiChunk, thereby validating the core value of fine-tuning for hierarchical chunking tasks. We use Qwen3-8B to generate response. The supplementary experimental results are presented in Table 6.

Table 6: Comparison of Chunking Accuracy and End-to-End RAG Performance: Few-Shot Prompting (1-shot/3-shot) vs. Fine-Tuned HiChunk on HiCBench.

Metric	**Method**		
	HC$_{1-shot}$	HC$_{3-shot}$	HC$_{ft}$
	Chunking Accuracy		
$F1_{L1}$	0.1784	0.2500	**0.4841**
$F1_{L2}$	0.1128	0.1203	**0.3140**
$F1_{ALL}$	0.2328	0.2199	**0.5450**
	RAG Performance (w/o Auto-Merge)		
ERec	72.35	72.26	**77.87**
Rouge	33.14	33.08	**35.21**
Fact-Cov	44.02	43.97	**46.84**
	RAG Performance (w Auto-Merge)		
ERec	73.47	73.05	**80.86**
Rouge	34.53	34.04	**36.17**
Fact-Cov	46.62	46.55	**49.97**

The experimental results demonstrate that increasing the number of few-shot examples did not improve the model's performance in chunking accuracy or the full-link performance of the subsequent RAG pipeline. Furthermore, all few-shot schemes show a significant performance gap compared to the fine-tuned HiChunk method. This fully confirms that fine-tuning is a necessary prerequisite for achieving high-quality hierarchical chunking of HiChunk.

6 Conclusion

This paper begins by analyzing the shortcomings of current benchmarks used for evaluating RAG systems, specifically highlighting how evidence sparsity makes them unsuitable for assessing different chunking methods. As a solution, we introduce HiCBench, a QA benchmark focused on hierarchical document chunking, which validly evaluates the impact of various chunking methods on the entire RAG process. Additionally, we propose the HiChunk framework, which, when combined with the Auto-Merge retrieval algorithm, significantly enhances the quality of chunking, retrieval, and model responses compared to other baselines.

7 Limitations

While the HiChunk framework theoretically supports the prediction and processing of documents with more than three hierarchical levels, the performance gain from high-level structures exhibits a trend of diminishing marginal returns in practical scenarios. This is primarily constrained by two universal challenges faced by all hierarchical chunking methods:

1. Data scarcity, as publicly available training data for documents with high hierarchical levels is extremely limited.
2. The funnel effect, where high-level modeling relies on the prediction accuracy of lower levels, leading to decreasing accuracy guarantees as the hierarchy deepens.

These two points do not indicate that HiChunk is unable to handle high-level document structures, but rather reflect universal challenges faced by all hierarchical chunking methods. In fact, even under the no-level-limit (LA) setting, HC200+AM still outperform those of other baseline methods (as shown in Table 3), fully demonstrating that the core value of HiChunk's hierarchical design is not affected by structures beyond three levels. How to model high-level structures more efficiently will be the focus of future research.

References

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Sinchana Ramakanth Bhat, Max Rudat, Jannis Spiekermann, and Nicolas Flores-Herr. 2025. Rethinking chunk size for long-document retrieval: A multi-dataset analysis. *arXiv preprint arXiv:2505.21700*.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024b. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Sangwoo Cho, Kaiqiang Song, Xiaoyang Wang, Fei Liu, and Dong Yu. 2022. Toward unifying text segmentation and long document summarization. *arXiv preprint arXiv:2210.16422*.

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. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

André V. Duarte, João DS Marques, Miguel Graça, Miguel Freire, Lei Li, and Arlindo L. Oliveira. 2024. LumberChunker: Long-form narrative document segmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6473–6486, Miami, Florida, USA. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Robert Friel, Masha Belyi, and Atindriyo Sanyal. 2024. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*.

Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao. 2024. Late chunking: contextual chunk embeddings using long-context embedding models. *arXiv preprint arXiv:2409.04701*.

Hangfeng He, Hongming Zhang, and Dan Roth. 2022. Rethinking with retrieval: Faithful large language model inference. *arXiv preprint arXiv:2301.00303*.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Arihant Jain, Purav Aggarwal, and Anoop Saladi. 2025. Autochunker: Structured text chunking and its evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, pages 983–995.

Jiajie Jin, Xiaoxi Li, Guanting Dong, Yuyao Zhang, Yutao Zhu, Yongkang Wu, Zhonghua Li, Ye Qi, and Zhicheng Dou. 2025. Hierarchical document refinement for long-context retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3502–3520, Vienna, Austria. Association for Computational Linguistics.

Tomáš Kočiský, 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.

Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. Text segmentation as a supervised learning task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 469–473, New Orleans, Louisiana. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Xianzhi Li, Samuel Chan, Xiaodan Zhu, Yulong Pei, Zhiqiang Ma, Xiaomo Liu, and Sameena Shah. 2023. Are chatgpt and gpt-4 general-purpose solvers for financial text analytics? a study on several typical tasks. *arXiv preprint arXiv:2305.05862*.

Yang Liu, Chenguang Zhu, and Michael Zeng. 2021. End-to-end segmentation-based news summarization. *arXiv preprint arXiv:2110.07850*.

Tengchao Lv, Lei Cui, Momcilo Vasilijevic, and Furu Wei. 2021. Vt-ssum: A benchmark dataset for video transcript segmentation and summarization. *arXiv preprint arXiv:2106.05606*.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and 1 others. 2021. Quality: Question answering with long input texts, yes! *arXiv preprint arXiv:2112.08608*.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and 1 others. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *arXiv preprint arXiv:2409.10173*.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, and 1 others. 2023. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*.

Zhitong Wang, Cheng Gao, Chaojun Xiao, Yufei Huang, Shuzheng Si, Kangyang Luo, Yuzhuo Bai, Wenhao Li, Tangjian Duan, Chuancheng Lv, and 1 others. 2025. Document segmentation matters for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8063–8075.

Haoqian Wu, Keyu Chen, Haozhe Liu, Mingchen Zhuge, Bing Li, Ruizhi Qiao, Xiujun Shu, Bei Gan, Liangsheng Xu, Bo Ren, and 1 others. 2023. Newsnet: A novel dataset for hierarchical temporal segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10669–10680.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. 2025. When to use graphs in rag: A comprehensive analysis for graph retrieval-augmented generation. *arXiv preprint arXiv:2506.05690*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 641–649, New York, NY, USA. Association for Computing Machinery.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018a. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018b. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Junyuan Zhang, Qintong Zhang, Bin Wang, Linke Ouyang, Zichen Wen, Ying Li, Ka-Ho Chow, Conghui He, and Wentao Zhang. 2024. Ocr hinders rag: Evaluating the cascading impact of ocr on retrieval-augmented generation. *arXiv preprint arXiv:2412.02592*.

Qinglin Zhang, Qian Chen, Yali Li, Jiaqing Liu, and Wen Wang. 2021. Sequence model with self-adaptive sliding window for efficient spoken document segmentation. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 411–418. IEEE.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, and 1 others. 2025. Siren's song in the ai ocean: A survey on hallucination in large language models. *Computational Linguistics*, pages 1–46.

Jihao Zhao, Zhiyuan Ji, Zhaoxin Fan, Hanyu Wang, Simin Niu, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2025. MoC: Mixtures of text chunking learners for retrieval-augmented generation system. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5172–5189, Vienna, Austria. Association for Computational Linguistics.

A Appendix

A.1 Statistics of Dataset

Table A1 presents detailed statistics of the four datasets employed in the experiments, covering core attributes of documents, question-answer (QA) pairs, and evidence segments. Specifically, the table includes the number of documents (Num_{doc}), average sentences per document (Sent_d), average words per document (Word_d), total QA pairs (Num_{qa}), average words of questions (Word_q) and answers (Word_a). The gray-highlighted rows further report the average word count (Word_e) and sentence count (Sent_e) of evidence segments for each dataset.

Table A1: Statistics of dataset used in experiments.

Dataset	**Qasper**	**GutenQA**	**OHRBench(T_0)**	**HiCBench(T_1, T_2)**
Num_{doc}	416	100	214	130
Sent_d	164	5,373	886	298
Word_d	4.2k	146.5k	26.8k	8.5k
Num_{qa}	1,372	3,000	4,702	(659, 541)
Word_q	8.9	16.0	22.2	(31.0, 33.0)
Word_a	16.0	26.0	4.8	(130.1, 126.4)
Word_e	239.4	39.3	39.1	(561.5, 560.5)
Sent_e	10.5	1.7	1.7	(20.5, 20.4)

A.2 Combination with Late-Chunking

In order to verify the complementarity of HiChunk with other optimization techniques, we supplemented the combination of Late-Chunking with various chunking methods and conducted experiments on HiCBench. The experiment setting is consistent with (Günther et al., 2024), using jina-embeddings-v3(Sturua et al., 2024) as the embedding model. The results are presented in Table A2.

Late-Chunking universally enhances the ERec and Fact-Cov metrics of various chunking methods. Regardless of whether Late-Chunking is integrated, HC200+AM consistently delivers the best performance across all evaluated settings. This result validates the flexibility of the HiChunk framework, whose design enables seamless integration with other RAG optimization techniques (e.g., Late-Chunking) to further boost end-to-end performance.

Table A2: The performance of combining the Late-Chunking and different chunking methods on HiCBench(T_1 and T_2). The best result is in **bold**, and the sub-optimal result is in underlined

Methods	**w/o Late-Chunking**			**w/ Late-Chunking**		
	ERec	Rouge	Fact-Cov	ERec	Rouge	Fact-Cov
C200	75.59	34.19	46.71	78.04	34.33	49.12
SC	73.07	34.17	45.60	78.07	34.16	48.45
LC	77.89	34.84	49.16	79.93	35.16	50.65
HC200	78.13	34.93	48.03	79.29	34.84	49.77
HC200+AM	**80.87**	**36.34**	**51.49**	**81.20**	**36.00**	**52.71**

A.3 Sensitivity Analysis on Threshold θ^* of $Cond_2$

The physical meaning of θ^* in $Cond_2$ is the minimum ratio of the total length of child nodes to the parent node length (controlling merging granularity). In order to verify the robustness of the Auto-Merge algorithm. We conduct the experiment by fixing θ from 0.0 to 1.0 (with an interval of 0.1) to test performance changes on three datasets. We use Qwen3-8B as generator model. The results are as Figure A1.

When θ ranges from 0.1 to 0.6, HiCBench's ERec remains above 80.05% and OHRBench's Rouge remains above 49.89%, indicating that the algorithm is robust to threshold variations; As θ increases, the performance of evidence-dense tasks decreases (ERec drops to 78.08% at θ=1.0), while the performance of evidence-sparse tasks improves when $\theta > 0.5$, reflecting the granularity demand differences between the two types of tasks; The adaptive threshold θ^* achieves cross-task balance through dynamic adjustment: it maintains high evidence recall on HiCBench (80.66%) while achieving the optimal Score (45.00) on LongBench and Rouge (50.00) on OHRBench. This proves that it can adapt to different tasks without manual parameter tuning, with better robustness than fixed thresholds.

A.4 Evidence Recall under Different Token Budget

In this section, we further present the curve of evidence recall metric at different retrieval context

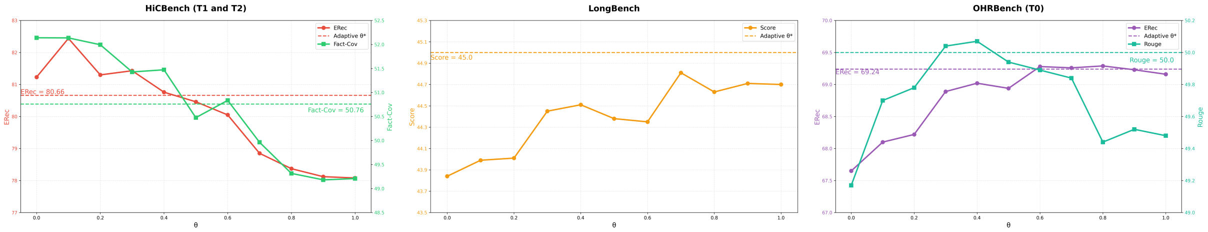


Figure A1: Performance changes across different θ on HiCBench, LongBench, and OHRBench. Dashed line represents the result of the adaptive θ^*.

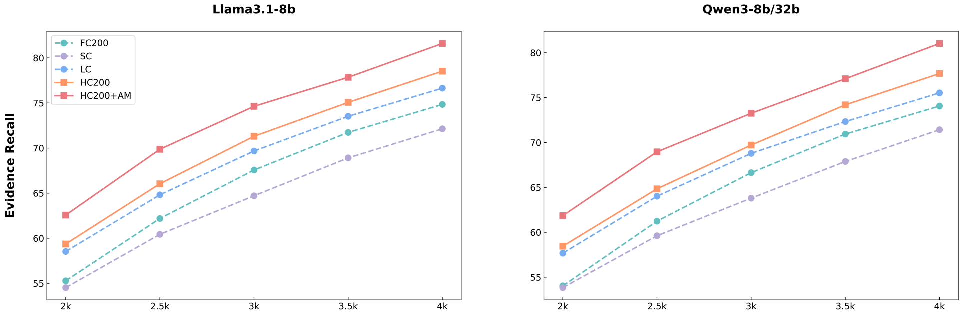


Figure A2: Evidence recall metric across different token budget on HiCBench(T_1).

length settings (from 2k to 4k). The results are shown in Figure A2. Compared with other chunking methods, the HC200+AM method always maintains the best performance.

A.5 Time Cost for Chunking

As document chunking is essential for RAG systems, it must meet specific timeliness requirements. In this section, we analyze the time costs associated with different semantic-based chunking methods, as presented in Table A3. Although the SC method exhibits superior real-time performance, it consistently falls short in quality across various datasets compared to other baselines. However, the LC method demonstrates reasonably good performance, but its chunking speed is considerably slower than other semantic-based methods, limiting its applicability within RAG systems. In contrast, the HC method achieves the highest chunking quality among all baseline methods while maintaining an acceptable time cost, making it well-suited for implementation in real scenarios.

Table A3: The average time cost on chunking documents.

Dataset	Avg. Word	SC		LC		HC	
		Time(s)	Chunks	Time(s)	Chunks	Time(s)	Chunks
Qasper	4,166	0.4867	43.83	5.4991	18.32	1.4993	15.08
Gov-report	13,153	1.3219	114.72	15.4321	40.89	4.3382	29.79
OHRBench	26,808	3.0943	249.14	37.3935	89.68	14.5776	92.23
GutenQA	146,507	16.5028	1,453.00	132.4900	393.52	60.1921	232.85
HiCBench	8,519	1.0169	80.12	13.4414	41.48	5.7506	51.35

A.6 HiCBench Annotation

The HiCBench dataset is derived from OHRBench, including documents in fields like Academic, Finance, and Administration. Documents with over 4,000 words were chosen as the initial corpus. Annotators read and comprehended the semantic structure of each document, then annotated its hierarchical structure. Following the same format as HiChunk training, they marked the positions of chunking points at different levels. The key criterion was "semantic independence" — ensuring each segmented unit could express a complete meaning on its own. Hierarchy levels were determined by identifying subordinate relationships. Each document was annotated at least twice. Only annotations with over 80% agreement on chunking point positions were retained to ensure data consistency.

A.7 Prompts Used

Listing A1: Prompt for segment summarization.

~~~
**Task:**
You are tasked with analyzing the provided document sections and their hierarchical
    structure. Your goal is to generate a concise and informative paragraph
    describing the content of each section and subsection.

**Instructions:**
1. Each section or subsection is identified by a header in the format `===SECTION
    xxx===` (for example, `===SECTION 1===`, `===SECTION 2.1===`, etc.).
2. For every section and subsection, write a brief, clear, and informative
    paragraph summarizing its content. Do not omit any section or subsection.
3. Present your output as a JSON object with the following structure:
```json
{
    "SECTION 1": "description of section 1",
    "SECTION 1.1": "description of section 1.1",
    ...
    "SECTION n.m": "description of section n.m"
}
~~~
4. Ensure that each key in the JSON object matches the exact section identifier
    (e.g., `"SECTION 2.1.3"`), and do not include any sections or subsections that
    are not present in the provided document fragment.
5. Do not add any commentary or explanation outside the JSON object.

Document Fragment:
```

Listing A2: Prompt for QA cunstruction.

```
You are provided with a document that includes a detailed structure of sections and
    subsections, along with descriptions for each. Additionally, complete contents
    are provided for a few selected sections. Your task is to create a question and
    answer pair that effectively captures the essence of the selected sections.
    Finally, you need to extract the facts which are mentioned in the answer.

<Type of Generated Q&A Task: Evidence-dense Dependent Understanding task>
Understanding task means that, the generated question-answering pairs that require
    the responser to extract information from documents. The answer should be able
    to find directly in the documents without any reasoning.
Evidence-dense dependent means that the facts about generated question are wildly
    distributed across all parts of the retrieved sections.

<Criteria>
- The question MUST be detailed and be based explicitly on information in the
    document.
- The question MUST include at least one entity.
- Question must not contain any ambiguous references, such as 'he', 'she', 'it',
    'the report', 'the paper', and 'the document'. You MUST use their complete
    names.
- The context sentence the question is based on MUST include the name of the
    entity. For example, an unacceptable context is "He won a bronze medal in the 4
    * 100 m relay". An acceptable context is "Nils Sandstrom was a Swedish sprinter
    who competed at the 1920 Summer Olympics."
- **THE MOST IMPORTANT: Evidence-dense dependency**, Questions must require
    understanding of ENTIRE selected sections. Never base Q&A on isolated few
    sentences. For example, a question comply the **Evidence-dense dependency**
    criteria means that the facts about this question should be wildly distributed
    across all parts of the retrieved sections.

<Output Format>
Your response should be structured as follows:
```json
{{
    "question": "Your generated question here",
    "answer": "Your generated answer here"
}}
```

<Document Structure and Description>
```
```

````
{section_description}

<Retrieved Section and Content>
{section_content}
````

Listing A3: Prompt for evidence retrieval.

~~~
**Task:**
Analyze the relationship between context sentences and answer sentences.

**Instructions:**
1. You are given:
    - A context fragment, with each sentence numbered as follows: `[serial number]:
    context sentence content`
    - A question and its corresponding answer, with each answer sentence numbered
    as follows: `<serial number>: answer sentence content`
2. For each sentence in the answer, identify which sentence(s) from the context
    provide the information used to construct that answer sentence.
3. Present your findings in the following JSON format:
```json
{{
    "<answer_sentence_id_1>": "[context_sentence_id_1], ...,
    [context_sentence_id_n]",
    "<answer_sentence_id_2>": "[context_sentence_id_1], ...,
    [context_sentence_id_m]",
    ...
    "<answer_sentence_id_i>": "[context_sentence_id_1], ...,
    [context_sentence_id_j]"
}}
```

**Notes:**
- Only include answer sentences that have supporting evidence in the context.
- If an answer sentence does not have a source in the context, do not include it in
    the JSON output.
- Use only the serial numbers (not the full sentences) for both context and answer
    sentences in your JSON output.
- If multiple context sentences support an answer sentence, list all relevant
    context sentence numbers, separated by commas.

**Context Sentences:**
{context_sentence_list}

**Question:**
{question}

**Answer Sentences:**
{answer_sentence_list}
~~~

Listing A4: Prompt for model training.

````
You are an assistant good at reading and formatting documents, and you are also
    skilled at distinguishing the semantic and logical relationships of sentences
    between document context. The following is a text that has already been divided
    into sentences. Each line is formatted as: "{line number} @ {sentence
    content}". You need to segment this text based on semantics and format. There
    are multiple levels of granularity for segmentation, the higher level number
    means the finer granularity of the segmentation. Please ensure that each Level
    One segment is semantically complete after segmentation. A Level One segment
    may contain multiple Level Two segments, and so on. Please incrementally output
    the starting line numbers of each level of segments, and determine the level of
    the segment, as well as whether the content of the sentence at the starting
    line number can be used as the title of the segment. Finally, output a list
    format result, where each element is in the format of: "{line number}, {segment
    level}, {be a title?}".

>>> Input text:
````

Algorithm 1: iterative inference

```
input  : Document $D$, Input length $L$
output : Global chunk points $GCP_{1:k}$
$S[1:N] \leftarrow$ SentTokenize($D$);
$a \leftarrow 1$;
$b \leftarrow \text{argmax}_{\hat{b}}(S[a:\hat{b}] \leq L)$;
$res\_lines \leftarrow$ None;
$GCP_{1:k} \leftarrow [] * k$ ;
while $1 \leq a < b \leq N$ do
    $LCP_{1:k} \leftarrow$ HiChunk($S[a:b]$, $res\_lines$);
    $GCP_{1:k} \leftarrow$ Merge($GCP_{1:k}$, $LCP_{1:k}$);
    if $len(LCP_1) \geq 2$ then
        $a \leftarrow LCP_1[-1]$;
        $res\_lines \leftarrow$ None;
    else
        $a \leftarrow b$;
        $res\_lines \leftarrow$ ResLines ($GCP_{1:k}$);
    $b \leftarrow \text{argmax}_{\hat{b}}(S[a:\hat{b}] \leq L)$;
return $GCP_{1:k}$
```

Algorithm 2: retrieval algorithm

```
input  : Token budget $T$, Chunks $C_{[1:M]}$, Query $q$
output : Retrieval context ctx
$C_{[1:M]}^{sorted} \leftarrow$ Sorted($C_{[1:M]}, q$);
$\mathcal{N} \leftarrow []$, $T_{used} \leftarrow 0$;
for $i \leftarrow 1$ to $M$ do
    $\mathcal{N} \leftarrow \mathcal{N} + C_{[i]}^{sorted}$;
    ctx, $T_{used} \leftarrow$ Context($\mathcal{N}$);
    $p \leftarrow$ parent($C_{[i]}^{sorted}$);
    while $Cond_{[1,2,3]}$ do
        if $T_{used} \geq T$ then
            break
        $\mathcal{N} \leftarrow$ Merge($\mathcal{N}$, $p$);
        ctx, $T_{used} \leftarrow$ Context($\mathcal{N}$);
        $p \leftarrow$ parent($p$);
    if $T_{used} \geq T$ then
        break
return ctx [: $T$]
```