From Context to EDUs: Faithful and Structured Context Compression via Elementary Discourse Unit Decomposition

Anonymous ACL submission

Abstract

Managing long context is a bottleneck for large language models (LLMs), introducing high costs and noise. Existing compression methods often disrupt local coherence or rely on latent encodings subject to positional bias and API incompatibility. We propose the EDU-based Context Compressor, an explicit framework designed to preserve structure and detail. We reformulate compression as a structure-then-select process: First, **LingoEDU** parses text into an Elementary Discourse Unit (EDU) relation tree, anchored to source indices to prevent hallucination. Second, a lightweight module ranks and selects query-relevant sub-trees. To evaluate structural understanding, we introduce **StructBench**, a manually annotated dataset of 248 documents. Empirical results demonstrate that our method achieves SOTA structural prediction accuracy and significantly outperforms frontier LLMs while reducing costs. Furthermore, our structure-aware compression substantially enhances performance across downstream tasks ranging from long-context tasks to complex Deep Search scenarios.

1 Introduction

Large Language Models (LLMs) have achieved remarkable progress (OpenAI, 2023; DeepSeek-AI et al., 2025), enabling sophisticated applications like long-context QA (Si et al., 2025b), multi-document summarization (Kryściński et al., 2022), and deep research agents (Zhang et al., 2025a). However, these tasks demand extensive context memory, creating a critical bottleneck. The resulting lengthy inputs incur prohibitive computational costs and introduce noise, which significantly degrades model performance (Hua et al., 2025).

Context compression has emerged as a crucial solution to reduce input length while preserving semantics. **Explicit compression methods** (Jiang et al., 2023; Xu et al., 2024) reduce length by removing less important tokens or using abstractive summarization (Xu et al., 2023). However, operating on discrete tokens or rigid boundaries often disrupts local coherence. Furthermore, focusing solely on global information leads to the loss of structural nuances and fine-grained details. Conversely, **implicit compression methods** (Mu et al., 2023; Liu and Qiu, 2025) encode text into latent vectors for higher compression ratios. Yet, recent studies (Li et al., 2025a) reveal that these methods suffer from positional bias, often neglecting content at the beginning or middle of the context to focus on salient details. Moreover, their reliance on latent vectors or specialized post-training (Cheng et al., 2025; Wei et al., 2025) restricts flexibility, rendering them incompatible with closed-source APIs like GPT-4.1 (OpenAI, 2025a).

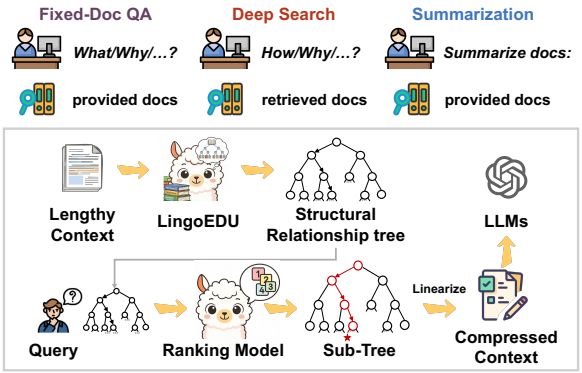


Figure 1: The illustration of our EDU-based Context Compressor framework. We formulate the context compression as a "Structure-then-Select" process.

In this work, we argue that an ideal compression strategy should be explicit and structure-aware to preserve both global foresight and fine-grained details. To this end, we introduce the EDU-based

Context Compressor, which reformulates compression as a structure-then-select process consisting of two modules: LingoEDU and a lightweight ranker. LingoEDU, inspired by Rhetorical Structure Theory (Mann and Thompson, 1988), transforms linear text into a relation tree of Elementary Discourse Units (EDUs)—variable-length units conveying coherent semantics. Unlike fixed tokens, EDUs allow nodes to be strictly anchored to source indices. We train LingoEDU via a novel human-in-the-loop pipeline followed by supervised fine-tuning. Subsequently, the ranking module identifies sub-trees relevant to the user query, filtering out structural noise. These selected sub-trees are then linearized into a compressed context. This explicit design ensures minimal hallucination, high coherence, and seamless compatibility with API-based models.

During the experiments, we aim to answer two pivotal questions: (1) Can advanced LLMs effectively compress long contexts while preserving original structural information? and (2) Does such structure-aware compression tangibly reduce the hallucinations for downstream tasks? To answer these questions, we first propose a manually annotated benchmark comprising 248 documents across diverse formats to evaluate the abilities of LLMs to understand and describe the structural information of the context. We find that even state-of-the-art models such as o3 (OpenAI, 2025b) still fail to fully understand the structural relationships within a given context, and prompt-based methods alone for these models struggle to effectively compress the context while preserving the original structural relations. Conversely, our well-trained EDU-based Context Compressor can effectively identify and preserve key structural elements while substantially reducing the context length. Meanwhile, we find that using the structured and compressed context from EDU-based Context Compressor not only reduces the input length for the model, but also filters out irrelevant and noisy tokens, thereby improving the model's performance on various tasks such as multi-document QA (Bai et al., 2024).

Our contributions are summarized as follows:

- **Novel Context Compression Framework:** We propose the EDU-based Context Compressor, utilizing document structure to create query-specific sub-trees. It preserves global and local details while remaining fully compatible with API-based models like GPT-4.1.
- **New Benchmark:** We construct a human-annotated benchmark comprising 248 documents across diverse formats to enable precise evaluation of the abilities of LLMs to understand structural information of the context.
- **SOTA Performance & Efficiency:** Empirical results show that our proposed LingoEDU outperforms advanced LLMs in understanding structural relations within the provided context and surpasses commercial APIs like Firecrawl, notably with a much lower cost.
- **Reducing Hallucinations for Long-context Tasks:** Experiments show that our proposed EDU-based Context Compressor can improve the performance and reduce hallucinations across diverse long-context tasks, e.g., multi-document QA, summarization, and search agent scenarios. For example, our method outperforms standard baselines (e.g., +14.94% on the HotpotQA dataset within LongBench). In Deep Search tasks, it also significantly enhances the performance, boosting DeepSeek-R1 by over **51.11%** relatively on the HLE.

2 Methodology

We propose the **EDU-based Context Compressor**, a novel framework designed to achieve faithful and structural-aware context compression. Implicit compression methods often suffer from positional bias and lack of transparency. Therefore, our approach adheres to an explicit compression paradigm: it first transforms the linear context sequence into a Structural Relation Tree, where semantic units are strictly anchored to the source text via coordinate pointers. As illustrated in Figure 1, the framework operates as a plug-and-play module compatible with any LLM (including API-based models). It consists of two cascaded components: the *LingoEDU*, which parses the document into discourse-connected units, and a *Ranking Module*, which identifies and linearizes the most relevant sub-trees to reconstruct a high-density context.

2.1 Overall Framework

Given a long input document $\mathcal{D}$ (or a set of documents) and a user query q, our goal is to overcome the limitations of fixed-size chunking and latent encoding. We reformulate the context compression task as a "Structure-then-Select" process:

Phase I: Structural Decomposition. The Decomposer transforms the unstructured linear text $\mathcal{D}$ into

a Structural Relation Tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$.

- **Nodes ($\mathcal{V}$):** represent **EDUs**, the minimal variable-length units capable of conveying coherent semantics. Crucially, strictly preserving the original text indices ensures hallucination-free hallucination.

- **Edges ($\mathcal{E}$):** represent the *discourse linkages* and dependency relations between EDUs (e.g., elaboration, contrast), capturing the logical flow often lost in standard retrieval.

Phase II: Sub-tree Retrieval and Linearization. A lightweight ranking module evaluates the semantic relevance between the query q and the structural nodes in $\mathcal{T}$. Instead of retrieving isolated sentences, it identifies the optimal task-relevant sub-trees $\mathcal{S} \subset \mathcal{T}$. Finally, these selected sub-trees are linearized back into a coherent text sequence $\mathcal{D}'$. This results in a compressed context where $|\mathcal{D}'| \ll |\mathcal{D}|$, retaining both global structural integrity and fine-grained details essential for complex downstream reasoning.

2.2 LingoEDU

In Phase I, we first use our proposed LingoEDU to extract the structural relation within the given context. As shown in Figure 2, our framework transforms a linear context sequence into a condensed hierarchical tree, where every node is strictly anchored to the source via coordinate pointers.

2.2.1 Problem Formulation

To enable this coordinate-based operation detailed in Figure 2(a), we first segment the input document $\mathcal{D}$ into a sequence of atomic building blocks termed **EDUs**. Formally, the document is represented as a sequence $\mathcal{U} = \{e_1, e_2, \ldots, e_N\}$, where each unit e_i constitutes a triplet:

$$e_i = (t_i, \texttt{pos}_i, \texttt{id}_i) \tag{1}$$

Here, t_i represents the textual content (typically a coherent clause or sentence), $\texttt{pos}_i$ denotes its physical grounding (character offsets), and crucially, $\texttt{id}_i$ acts as a unique sequential index. This index establishes a discrete Coordinate System, allowing the system to reference content by rigorous pointers. The core objective is to learn a decomposition function $f : \mathcal{E} \to \mathcal{T}$ that maps the linear EDU sequence to a latent hierarchical tree $\mathcal{T}$ (Figure 2(c)). Each node $n_j \in \mathcal{T}$ acts as a compressed semantic capsule defined as:

$$n_j = (h_j, l_j, \sigma_j), \quad \text{where } \sigma_j = [\texttt{id}_{\text{start}}, \texttt{id}_{\text{end}}] \tag{2}$$

In this tuple, h_j is the semantic abstract (e.g., a section title or summary), l_j denotes the hierarchical depth, and σ_j represents the **EDU Span**—a closed interval explicitly pointing to the source range in $\mathcal{E}$. By strictly enforcing the constraint $\sigma_j \subseteq [1, N]$, we ensure referential integrity: the generated structure is a lossless index purely derived from the input context, guaranteeing effectively eliminating extrinsic hallucinations.

2.2.2 Training Strategy and Data Synthesis

Since high-quality, fine-grained hierarchical annotations for long contexts are scarce, we introduce a scalable automated pipeline to synthesize training data. This pipeline leverages a strong LLM to distill the logic of summarization-and-indexing.

Bi-Level Task Decomposition. Correctly understanding the structure of the context equires different cognitive capabilities for different structural types, we decouple the data generation into two distinct sub-tasks to prevent "instruction conflict" (conflating visual layout with semantic reasoning): (1) *Explicit Layout Extraction:* The model extracts objective structural cues (e.g., Markdown headers, HTML tags) to form the document skeleton. This task enables high certainty with low ambiguity. (2) *Deep Semantic Segmentation:* For large text blocks lacking explicit formatting, the model focuses purely on semantic shifts to delineate finer-grained functional sections. This requires deep reasoning to resolve high ambiguity.

The Solver-Critic Refinement Loop. To ensure the synthesized labels are high-quality, we implement an *Iterative Refinement Mechanism*:

- **The Solver:** Proposes a hierarchical decomposition, attempting to abstract detailed content into high-level semantic nodes.

- **The Critic:** Audits the proposal specifically checking whether the generated abstract (Title) accurately reflects the assigned span σ_j without hallucination or semantic drift.

This adversarial collaboration ensures that despite the high compression rate, the structural integrity of the training data remains intact.

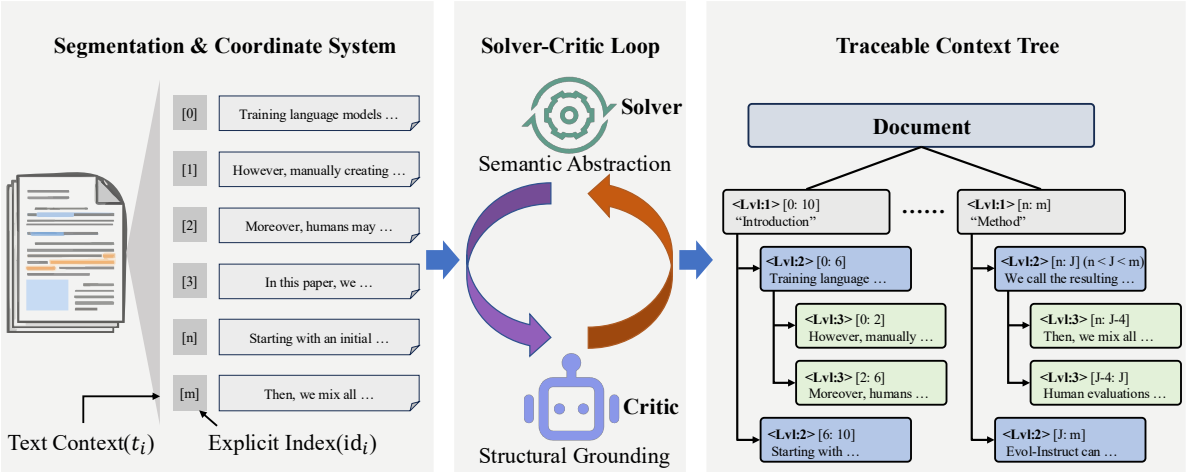


Figure 2: **Overview of the LingoEDU.** (a) The continuous input text is segmented into EDUs, creating an addressable sequence where each unit carries a unique coordinate ID. (b)Unlike standard summarization, the model outputs *Augmented Markdown*. It performs compression by generating closed index intervals (e.g., [12–15]) rather than regenerating body text, effectively indexing the content. (c) The output is parsed into a hierarchical semantic tree $\mathcal{T}$. The ID spans serve as unhallucinated anchors, ensuring nodes remain strictly faithful to the source context.

2.2.3 Traceable Generation

We train an open-source LLM on this synthesized corpus. To further guarantee robustness during inference, we employ an Augmented Markdown Schema (Figure 2(b)). The model is trained to generate nodes in the following format:

$$\text{Output} = \underbrace{\#\#}_{\text{Level}} \ \underbrace{[\text{id}_{\text{start}}\text{–id}_{\text{end}}]}_{\text{Traceable Anchor}} \ \underbrace{\text{Concept Title}}_{\text{Semantic Abstract}} \quad (3)$$

This design achieves dual goals: (1) Token Efficiency, as long text blocks spanning multiple EDUs are compressed into minimal tokens; and (2) Hallucination Elimination, as the strict span format forces the model to rely on the coordinate system rather than free-form generation. Constraints are further enforced during decoding to ensure only valid numerical indices from $\mathcal{E}$ are generated.

2.3 Ranking Module

Once the document is decomposed into the structural tree $\mathcal{T}$, the challenge shifts to identifying which nodes contribute most to answering the user query q. We introduce a *Budget-Aware Semantic Filter* that leverages the Decomposer's output to perform precise context selection.

2.3.1 Why Node-Level Ranking?

Instead of retrieving at the sentence level—which often results in context fragmentation—or the document level—which introduces excessive noise—we perform retrieval at the *Node Level*. Our nodes n_j encapsulate semantic completeness via their spans, allowing the model to judge the relevance of larger logical blocks without processing the full text.

2.3.2 Plug-and-Play Relevance Scoring

We define a relevance scoring function $\phi(q, n_j)$ to quantify the pertinence of each node. While our framework allows ϕ to be instantiated by LLMs (e.g., GPT-4) via prompt-based scoring, such approaches are computationally prohibitive for scanning dense tree structures. To balance performance with efficiency, we employ a lightweight ranking model (Zhang et al., 2025b) as a cost-effective surrogate. Specifically, we utilize an open-source model to compute the relevance score s_j:

$$s_j = \phi_\theta \left(q,\ h_j \oplus t_{\text{rep}}\right) \quad (4)$$

where h_j is the generated abstract (title) and t_{rep} is the representative text snippet from the span σ_j. By using a compact model (e.g., 0.6B parameters) rather than large-scale LLM APIs, we achieve high-throughput filtering with negligible latency.

2.3.3 Budget-Aware Greedy Selection

To address the limitations of fixed Top-K retrieval (which neglects variable token consumption), we employ a dynamic adaptive selection strategy bounded by a context budget B_{max}.

We sort all nodes $n_j \in \mathcal{T}$ by their score s_j in descending order and iteratively accumulate nodes into a candidate set $\mathcal{C}$:

$$\mathcal{C} = \left\{ n_j \mid \sum_{n \in \mathcal{C}} \text{Len}(\text{Retrieve}(\sigma_n)) \leq B_{\text{max}} \right\} \quad (5)$$

where Retrieve(σ_n) fetches the original text EDUs corresponding to the span $[\texttt{id}_{\text{start}}, \texttt{id}_{\text{end}}]$. Details and ablations against leaf-only selection are provided in Appendix C.

2.3.4 Linearization

Standard RAG often loses discourse coherence when disjoint chunks are concatenated. Thanks to the explicit coordinates `id` provided by our Decomposer, we apply a re-ordering method. The selected spans in $\mathcal{C}$ are sorted by their original start indices $\texttt{id}_{\text{start}}$ before concatenation. This restoration of logical order enables the downstream LLM to perform effective reasoning across discontinuous but structurally organized segments.

3 Experiments

3.1 Evaluation of Structural Integrity and Compression

In this section, we address the first research question: *Can state-of-the-art LLMs effectively compress long contexts while preserving original structural information?* We compare our proposed method against frontier LLMs and commercial parsing APIs on a newly constructed benchmark.

3.1.1 Experiment Settings

Benchmark Construction. The absence of public benchmarks for fine-grained document structure analysis motivated us to construct a specific dataset named **StructBench**. We compiled a test set of 248 documents, covering diverse formats (Web pages, PDFs), languages (Chinese, English), and genres (e.g., government files, institutional reports, academic papers, and technical tutorials). The dataset spans 10 distinct genres, primarily focusing on complex structures such as academic papers, government files, business reports, and technical tutorials (full distribution in Appendix D). Document lengths vary significantly, ranging from 300 to 50,000 words. To ensure high-quality ground truth, documents were parsed, sentence-segmented, and manually annotated for discourse structure by human experts. To enable fair comparison with baselines that may struggle with leaf-level details, we extracted the *structural backbone* (top-level hierarchy) from the annotations to serve as the labels.

Evaluation Metrics. We employ two complementary metrics to evaluate structural fidelity. Tree Edit Distance (TED) (Zhang and Shasha, 1989) acts as a micro-level metric to measure structural dissimilarity by computing the minimum number of edit operations (insertion, deletion, substitution) required to transform the predicted tree into the ground truth, where a lower TED indicates more precise structural alignment. Complementarily, Document Level Accuracy (DLA) serves as a macro-level metric defined as $\text{DLA} = \frac{|D_{\text{match}}|}{|D_{\text{all}}|}$, in which D_{match} represents the count of documents where the decomposed structural backbone perfectly matches the ground truth. This constitutes a rigorous metric that requires zero structural errors.

Baselines. We compare our **LingoEDU** (built on Qwen3-4B (Team, 2025)) against two types of strong baselines. For LLMs, we designed specific prompts to query them to output JSON/Markdown structures: (1) Frontier LLMs: We evaluated SOTA models including GPT-4o/4.1 (Hurst et al., 2024), OpenAI o3/o4-mini (OpenAI, 2025b), Claude 3.7 Sonnet/4 (Anthropic, 2025), DeepSeek-V3/R1 (DeepSeek-AI et al., 2025), and Qwen3-235B (Yang et al., 2025). All results are averaged over three distinct runs. (2) Commercial Parsing APIs: We selected Jina Reader and Firecrawl, which are widely used for web-to-markdown conversion. We deployed test documents on a static server to allow URL-based access.

Implementation Details. Our method utilizes Qwen3-4B as the backbone. The training involved a two-stage process: (1) Continued Pre-training on $\sim$100k synthetic samples to learn layout patterns, followed by (2) Supervised Fine-Tuning (SFT) on thousands of manually annotated documents to align with human intent. All experiments were conducted on a Linux operating system running on a high-performance server equipped with an Intel Xeon 2.3GHz CPU, 1960GB of memory, and 8 NVIDIA A100 GPUs, each with 80 GB of VRAM.

3.1.2 Experiment Results

Analysis of Structural Integrity. Table 1 highlights the superiority of our explicit training paradigm. While top-tier commercial LLMs like Claude-4-Sonnet and OpenAI o3 achieve competitive structural scores (TED $\sim$5.1–5.5), other robust models such as DeepSeek-R1 and Qwen3 still struggle, plateauing at a TED of 6.2–9.9. Qualitative analysis reveals that general models often *hallucinate* non-existent sub-sections or flatten deep hierarchies to save generation tokens. Similarly, commercial parsing APIs lack semantic depth; Jina and Firecrawl exhibit high TED scores (> 16) as they rely on shallow HTML tags and fail to capture implicit discourse structures found in complex

Method	Type	TED (Structure) ↓	DLA (Accuracy) ↑	Cost ($) ↓	Latency (s) ↓
GPT-4o	General LLM*	6.22	29.03%	5.21	-
GPT-4.1		6.35	37.90%	4.17	-
OpenAI o3		5.51	28.63%	4.17	-
OpenAI o4-mini		5.87	32.66%	2.28	-
Claude-3.7-Sonnet		6.65	35.08%	7.09	-
Claude-4-Sonnet		<u>5.08</u>	<u>43.15%</u>	7.09	-
Gemini-2.5-Flash		5.82	27.82%	0.99	-
Gemini-2.5-Pro		5.61	32.66%	4.02	-
DeepSeek-V3		6.32	33.47%	0.30	-
DeepSeek-R1		6.26	30.65%	1.14	-
Qwen3-32B		9.49	24.90%	0.26	10.17†
Qwen3-235B		9.93	17.89%	0.11	-
Jina-Reader	Parser API	17.04	-	**0.10**	-
Firecrawl		16.81	-	<u>0.17</u>	-
Our Method (LingoEDU)	**Specialized**	**4.77**	**49.60%**	<u>0.17</u>	**1.20†**

Table 1: Performance comparison on **StructBench**. ∗ indicates the model is accessed via API. † denotes local deployment for latency testing using equivalent hardware. Costs are calculated for the entire test set (approx. 248 documents). Best results are **bolded**, and second-best are <u>underlined</u>.

PDFs. In contrast, our LingoEDU Decomposer demonstrates specialized efficiency by achieving a remarkable TED of **4.77** and a DLA of **49.60%**, significantly outperforming the strongest baseline, Claude-4-Sonnet (+**6.45%** absolute DLA). This confirms that structural understanding requires dedicated supervision beyond what emergent prompting or reasoning models can provide.

Efficiency and Cost Analysis. In real-world long-context applications, overhead is critical. As shown in Table 1, our method offers an optimal trade-off. It matches the cost of the cheapest parsers ($0.0007/doc) while delivering a latency of just 1.20 seconds per document—nearly **10× faster** than a locally deployed Qwen3-32B. This efficiency stems from our architecture's design, which outputs compact coordinate indices instead of generating verbose text.

Ablation Setting	Variant	TED	DLA (%)
Output Formulation	Indices Only	8.16	33.06
	Indices + Text (Ours)	**4.77**	**49.60**
Data Scaling	20% Data	4.87	45.16
	50% Data	4.85	48.79
	100% Data	**4.77**	**49.60**

Table 2: Ablation Studies. We analyze the impact of formulation and training data scale.

Model Size	TED (↓)	DLA % (↑)
Qwen-1.7B	4.99	48.39
Qwen-4B	**4.77**	**49.60**
Qwen-8B	4.89	49.19

Table 3: Ablation study on backbone model scaling. Our 4B model achieves the best balance between structural error (TED) and relation accuracy (DLA).

3.1.3 Ablation Studies

Table 2 validates the effectiveness of our design choices. First, the significant performance gap between "Indices Only" and our method (33.06% vs. 49.60% DLA) highlights that explicit text generation acts as a crucial semantic anchor for structural prediction. Second, the model exhibits remarkable data efficiency; even when scaled down to just **20%** of the training data, it achieves a TED of 4.87 and retains over **91%** of the full model's accuracy.

Finally, we investigate the impact of model scale on structural parsing capability, as shown in Table 3. Scaling the backbone from 1.7B to 4B yields clear improvements, reducing TED from 4.99 to 4.77. However, further scaling to 8B results in performance saturation: both TED and DLA slightly regress compared to the 4B model. This suggests that the 4B parameter range strikes an optimal balance for this specific task, whereas larger models may suffer from overfitting to the rigid output format without proportionally larger datasets.

3.2 Main Results on Downstream Long-Context Tasks

In this section, we address the second research question: *Does structure-aware compression tangibly enhance performance for downstream tasks?* We evaluate the utility of our EDU-based Context Compressor across two distinct scenarios: standard long-context benchmarks and complex open-domain Deep Search pipelines.

Model / Method	Multi-Doc QA				Summarization				Few-shot			
	HotpotQA	2Wiki	Musique	DuReader	GovRep	QMSum	MultiN	VCSum	TREC	Trivia	SAMSum	LSHT
C3	0.07	0.09	0.08	2.08	18.20	7.35	18.03	0.39	1.00	6.42	8.29	6.50
Glyph	66.42	72.98	-	-	25.53	19.78	-	-	**82.62**	88.54	-	-
Gemini-2.5-Pro												
Standard	35.20	38.10	28.55	7.15	4.10	15.80	4.05	5.80	46.50	59.85	20.45	26.10
Self-Sum	37.78	39.90	30.77	7.79	4.34	16.53	4.44	6.17	49.00	62.31	21.89	29.50
LLMLingua-2	24.84	22.31	21.11	5.55	3.94	12.95	4.71	5.58	22.12	55.37	19.15	16.13
Ours (LingoEDU)	40.46	40.91	31.22	8.12	4.25	16.17	4.85	6.36	57.50	63.25	23.80	35.48
Δ (vs. Standard)	*+14.94%*	*+7.38%*	*+9.35%*	*+7.69%*	*+2.44%*	*+2.34%*	*+19.75%*	*+9.66%*	***+23.66%***	*+1.25%*	*+11.39%*	*+3.45%*
GPT-4.1												
Standard	65.83	72.98	51.90	21.80	29.97	22.84	20.85	12.50	77.00	90.07	39.20	48.60
Self-Sum	67.89	74.39	53.48	23.51	30.98	22.53	22.06	13.71	79.00	93.69	40.79	50.50
LLMLingua-2	65.90	**75.42**	54.65	22.13	30.30	20.48	23.15	13.54	25.50	86.44	37.83	18.25
Ours (LingoEDU)	**70.11**	74.68	**54.86**	**25.34**	**31.56**	**23.30**	**23.50**	**14.62**	80.00	**93.76**	**41.68**	**52.50**
Δ (vs. Standard)	*+6.50%*	*+2.33%*	*+5.70%*	*+16.24%*	*+2.94%*	*+0.61%*	*+5.80%*	*+8.96%*	*+3.90%*	*+4.10%*	*+6.33%*	*+8.02%*

Table 4: Results on **LongBench**. Datasets are grouped by task type (columns). Δ denotes the relative improvement of **Ours** over the Standard baseline for each specific backbone. **Bold** indicates best performance per backbone group; underlined indicates second-best.

3.2.1 General Long-Context Understanding

We evaluate on LongBench (Bai et al., 2024), a benchmark covering multi-document QA, summarization, and few-shot learning.

Baselines. We compare three configurations: (1) Standard, which feeds the full (or truncated) original context directly to the LLM; (2) Self-Sum, which utilizes the LLM itself to generate abstractive summaries of the context prior to processing; (3) Glyph (Cheng et al., 2025), which applies an implicit compression baseline to serve as a reference for latent-based methods; and (4) C3 (Liu and Qiu, 2025), which employs a small LLM to aggressively compress long contexts into compact latent representations before decoding with a LLM; and (5)LLMLingua-2 (Jiang et al., 2023), a compression method formulated as token-level classification, distilled from an LLM and implemented with a bidirectional Transformer encoder to retain crucial information while improving efficiency.

Analysis of Results. Table 4 compares performance across Gemini-2.5-Pro and GPT-4.1 backbones. Overall, our EDU-based approach yields consistent improvements over the Standard baseline, with relative gains (Δ) peaking at **+23.66%** on few-shot tasks. Regarding Multi-Document QA (HotpotQA, Musique), our method outperforms Self-Sum. We attribute this to the tendency of abstractive methods to lose critical entities required for multi-hop reasoning; in contrast, our explicit tree structure preserves precise evidence chains via original text indices, enabling the model to look up exact details without hallucination. As for Summarization Tasks, while Self-Sum is naturally strong, our method remains competitive. Notably, it significantly outperforms the external Glyph baseline, suggesting that retaining hierarchical structure is more effective than latent compression for information retention.

Ablation Studies on Node Ranking Strategies. To isolate the impact of the ranking mechanism within LingoEDU, we decouple node selection from the reasoning backbone. We fix the generator as GPT-4.1 across all experiments and compare five configurations: (1) Standard: Feeds the full (or truncated) context directly without explicit filtering. (2) Random: Randomly selects nodes to match our compression budget, serving as a stochastic lower bound. (3) BM25: A sparse retrieval baseline relying on lexical overlap. (4) Self-Sum: Prompts the generator (GPT-4.1) itself to identify relevant nodes prior to reasoning. (5) Ours (LingoEDU): Semantically scores nodes using our lightweight Qwen3-Reranker-0.6B.

Table 5 demonstrates that our dedicated ranking approach consistently outperforms other strategies. While the Standard baseline is hindered by noise in long contexts, structured selection significantly boosts performance. Crucially, Ours surpasses Self-Sum across most datasets. This result underscores that a specialized, lightweight dense ranker (0.6B)—despite its size—offers superior evidence localization compared to the intrinsic selection capabilities of a general-purpose LLM, which often lacks the granularity for precise context pruning. Furthermore, the substantial margin over BM25 validates the necessity of semantic-aware filtering over surface-level matching.

We further validate the effectiveness and efficiency of our framework through extensive supple-

Selection Strategy	Multi-Doc QA				Summarization				Few-shot			
	HotpotQA	2Wiki	Musique	DuReader	GovRep	QMSum	MultiN	VCSum	TREC	Trivia	SAMSum	LSHT
Standard (No Selection)	65.83	72.98	51.90	21.80	29.97	22.84	20.85	12.50	77.00	90.07	39.20	48.60
Random	56.42	45.11	37.71	21.70	30.52	20.85	22.38	12.53	31.50	91.81	38.13	27.50
BM25	65.99	59.91	48.71	**26.84**	31.43	23.16	23.05	12.59	53.00	91.58	37.65	36.75
Self-Sum (LLM-Select)	67.89	74.39	53.48	23.51	30.98	22.53	22.06	13.71	79.00	93.69	40.79	50.50
Ours (Qwen3-Reranker 0.6B)	**70.11**	**74.68**	**54.86**	25.34	**31.56**	**23.30**	**23.50**	**14.62**	80.00	**93.76**	**41.68**	**52.50**

Table 5: Ablation study on ranking models. All methods use GPT-4.1 as the final generator. We compare No Selection, Random, BM25, LLM Self-Selection, and Our Dedicated Reranker. **Bold** denotes the best performance.

Model Backbone	HLE				BrowseComp-ZH			
	Base	Self-Sum	**Ours (LingoEDU)**	Δ	Base	Self-Sum	**Ours (LingoEDU)**	Δ
DeepSeek-R1	9.0	9.5	**13.6**	+51.11%	18.7	19.4	**20.4**	+9.09%
Qwen3-235B-Thinking	14.2	14.7	**15.5**	+9.15%	8.7	9.0	**12.8**	+47.13%
DeepSeek-V3.1	14.5	14.8	**15.6**	+7.59%	29.1	29.8	**38.8**	+33.33%
DeepSeek-V3.2	20.0	20.6	**21.2**	+6.00%	31.1	32.2	**34.6**	+11.25%
Closed-Source Models								
GPT-5	25.0	25.9	**27.1**	+8.40%	29.1	29.8	**31.8**	+9.28%
Claude Opus 4.1	14.0	14.8	**15.5**	+10.71%	20.8	21.5	**23.2**	+11.54%
Gemini 3 Pro (w/o Deep Think)	26.1	26.7	**30.1**	+15.33%	47.4	48.1	**48.8**	+2.95%

Table 6: Ablation study of the LingoEDU module on Deep Search benchmarks. Accuracy scores (%) are reported. Base: Standard Deep Search without compression; Self-Sum: Query-focused summarization; Ours (LingoEDU): Structural decomposition. Δ denotes the relative improvement of Ours over the Base baseline.

mentary analyses in Appendix E.

3.2.2 Impact on Deep Search

Unlike standard retrieval, Deep Search involves aggregating information from diverse, often noisy web sources to answer complex, open-ended queries. We integrated the LingoEDU module into a Deep Search pipeline and evaluated it on two challenging benchmarks(HLE (Phan et al., 2025) and BrowseComp-ZH (Zhou et al., 2025)) designed to push the limits of current LLMs.

Robustness to Web Noise. As shown in Table 6, on the *BrowseComp-ZH*, our method delivers dramatic gains. Specifically, Qwen3-235B and DeepSeek-V3.1 show relative improvements of **47.13%** and **33.33%**, respectively. This confirms that our Decomposer acts as a semantic filter: by identifying logical EDUs, it effectively prunes "structural dead branches" while preserving the core content, a capability that standard summarization lacks in such high-entropy Chinese web contexts.

Scaffolding for Complex Reasoning. For the academic reasoning required by *HLE*, where answers depend on synthesizing multiple distant clues across disciplines, DeepSeek-R1 gains **+51.11%** relatively (from 9.0 to 13.6). This suggests that providing a cleaner, structure-aware context acts as a "reasoning scaffold." It prevents the model from getting lost in irrelevant details (context drift), allowing it to focus its compute budget on logical deduction across valid evidence.

Compatibility with Frontier Models. Critically, our method remains additive even for the most advanced models like GPT-5 and Gemini 3 Pro (w/o Deep Think)(Table 6, rows 4-6). This indicates that even as model capacity grows, handling unstructured noise and long-context reasoning remain bottlenecks. Our explicit compression provides a complementary signal that enhances the internal processing of SOTA LLMs.

4 Conclusion

In this work, we present the EDU-based Context Compressor, a plug-and-play framework bridging the gap between extended context windows and effective reasoning. By pivoting from linear reduction to hierarchical, discourse-aware decomposition, our method effectively filters noise while preserving the evidence chains required for complex tasks. We further validate this approach via StructBench, revealing that even state-of-the-art generalist LLMs lack the fine-grained structural analysis capabilities of our specialized model. Extensive experiments demonstrate that our method outperforms compression baselines and acts as a robust reasoning scaffold for search agents in noisy environments. These findings identify explicit structural modeling as a critical prerequisite for advanced long-context understanding. Future work will extend this paradigm to multi-modal contexts and dynamic agent memory.

Limitations

Our EDU-based Context Compressor is currently text-only and tailored to formally structured documents, making it less reliable for highly informal or noisy inputs (e.g., social media, code-switched dialogue, low-quality OCR), and its effectiveness depends on accurate EDU segmentation and a lightweight discourse relation inventory that may miss subtle argumentative structures; the budget-aware, query-conditioned selection can prune indirectly relevant evidence under tight token budgets and is sensitive to reranker calibration across domains and languages, while index-anchored abstracts can still drift semantically and faithfully preserve misleading or outdated spans in the source; moreover, evaluations emphasize structural fidelity (TED, DLA) and a subset of long-context tasks, leaving interactive agent settings underexplored, and scaling to book-length inputs stresses latency and memory—future work will expand language coverage, strengthen robustness (adversarial and contradiction-aware), enrich discourse relations, improve selection objectives, address privacy concerns, and extend the framework to multimodal co-structuring of visual and textual tokens.

References

Anthropic. 2025. Introducing claude 4.

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.

Jiale Cheng, Yusen Liu, Xinyu Zhang, Yulin Fei, Wenyi Hong, Ruiliang Lyu, Weihan Wang, Zhe Su, Xiaotao Gu, Xiao Liu, et al. 2025. Glyph: Scaling context windows via visual-text compression. *arXiv preprint arXiv:2510.17800*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, Singapore. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2025. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 6491–6501.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. 2024.

Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*.

Qishuo Hua, Lyumanshan Ye, Dayuan Fu, Yang Xiao, Xiaojie Cai, Yunze Wu, Jifan Lin, Junfei Wang, and Pengfei Liu. 2025. Context engineering 2.0: The context of context engineering. *Preprint*, arXiv:2510.26493.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2024. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.* Just Accepted.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Xuhui Jiang, Yuxing Tian, Fengrui Hua, Chengjin Xu, Yuanzhuo Wang, and Jian Guo. 2024. A Survey on Large Language Model Hallucination via a Creativity Perspective. *Preprint*, arXiv:2402.06647.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. 2022. Booksum: A collection of datasets for long-form narrative summarization. *Preprint*, arXiv:2105.08209.

Yangning Li, Shaoshen Chen, Yinghui Li, Yankai Chen, Hai-Tao Zheng, Hui Wang, Wenhao Jiang, and Philip S. Yu. 2025a. Admtree: Compressing lengthy context with adaptive semantic trees. *Preprint*, arXiv:2512.04550.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6342–6353, Singapore. Association for Computational Linguistics.

Zongqian Li, Yixuan Su, and Nigel Collier. 2025b. 500xcompressor: Generalized prompt compression for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25081–25091.

Fanfan Liu and Haibo Qiu. 2025. Context cascade compression: Exploring the upper limits of text compression. *Preprint*, arXiv:2511.15244.

Hao Liu, Yu Lei, and Zhen Wu. 2025. Prosocial behavior in large language models: Value alignment and affective mechanisms. *Science China Technological Sciences*, 68(8):1820403.

William C Mann and Sandra A Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text-interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

Jesse Mu, Xiang Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

OpenAI. 2025a. Introducing gpt-4.1 in the api.

OpenAI. 2025b. Openai o3 and o4-mini system card.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981, Bangkok, Thailand. Association for Computational Linguistics.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Shuzheng Si, Haozhe Zhao, Gang Chen, Cheng Gao, Yuzhuo Bai, Zhitong Wang, Kaikai An, Kangyang Luo, Chen Qian, Fanchao Qi, et al. 2025a. Aligning large language models to follow instructions and hallucinate less via effective data filtering. *arXiv preprint arXiv:2502.07340*.

Shuzheng Si, Haozhe Zhao, Gang Chen, Yunshui Li, Kangyang Luo, Chuancheng Lv, Kaikai An, Fanchao Qi, Baobao Chang, and Maosong Sun. 2025b. GATEAU: Selecting influential samples for long context alignment. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7391–7422, Suzhou, China. Association for Computational Linguistics.

Shuzheng Si, Haozhe Zhao, Cheng Gao, Yuzhuo Bai, Zhitong Wang, Bofei Gao, Kangyang Luo, Wenhao Li, Yufei Huang, Gang Chen, et al. 2025c. Teaching large language models to maintain contextual faithfulness via synthetic tasks and reinforcement learning. *arXiv preprint arXiv:2505.16483*.

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

Haoran Wei, Yaofeng Sun, and Yukun Li. 2025. Deepseek-ocr: Contexts optical compression. *Preprint*, arXiv:2510.18234.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *Preprint*, arXiv:2310.04408.

Yang Xu, Yunlong Feng, Honglin Mu, Yutai Hou, Yitong Li, Xinghao Wang, Wanjun Zhong, Zhongyang Li, Dandan Tu, Qingfu Zhu, et al. 2024. Concise and precise context compression for tool-using language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 16430–16441.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262.

Wan Zhang and Jing Zhang. 2025. Hallucination mitigation for retrieval-augmented large language models: A review. *Mathematics*, 13(5).

Wenlin Zhang, Xiaopeng Li, Yingyi Zhang, Pengyue Jia, Yichao Wang, Huifeng Guo, Yong Liu, and Xiangyu Zhao. 2025a. Deep research: A survey of autonomous research agents. *arXiv preprint arXiv:2508.12752*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Haozhe Zhao, Shuzheng Si, Liang Chen, Yichi Zhang, Maosong Sun, Baobao Chang, and Minjia Zhang. 2025. Looking beyond text: Reducing language bias in large vision-language models via multimodal dual-attention and soft-image guidance. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 19677–19701, Suzhou, China. Association for Computational Linguistics.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, et al. 2025. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *arXiv preprint arXiv:2504.19314*.

A Appendix Outline

This appendix is organized as follows:

- In Section B, we discuss related work in context compression and hallucination mitigation to position our contribution.
- In Section C, we use a sample to visualize the transformation process of our traceable context compression method.
- In Section D, we provide detailed statistics and domain breakdowns for StructBench, LongBench, and DeepSearch datasets.
- In Section E, we list the detailed model specifications, ensuring reproducibility.
- In Section E.3 and E.5, we report additional analysis on robustness across context lengths and the impact of retrieval granularity.
- In Section E.4, we conduct a comprehensive cost analysis proving the computational and economic efficiency of our pipeline.
- In Section F, we present the full prompt templates used for baselines, structure parsing, and downstream tasks.

B Related Work

B.1 Context Compression

As the input context length for LLMs grows, compressing long documents into efficient and manageable representations has become a critical challenge. To address this, existing approaches can generally be categorized into semantic summarization, soft prompting, and token-level pruning.

Explicit Compression Methods. Dominant approaches in this category operate on discrete tokens, filtering out low-informative content to reduce computational overhead. Early methods like Selective-Context (Li et al., 2023) and LLMLingua (Jiang et al., 2023) utilize perplexity-based metrics to prune redundant tokens. Recent advances, such as LLMLingua-2 (Pan et al., 2024) and TokenSkip (Xia et al., 2025), move towards data distillation and controllable pruning for higher efficiency. While effective at reducing sequence length, these methods often operate on discrete tokens or rigid sentence boundaries, disrupting the local coherence of the text. Meanwhile, they typically focus on preserving the most important global information, while overlooking the original article's structural information and fine-grained details.

Implicit Compression Methods. Alternatively, implicit methods map long contexts into continuous vector spaces or latent states. Works like AutoCompressor (Chevalier et al., 2023) and ICAE (Ge et al., 2023) compress text segments into soft prompts or memory slots. More extreme approaches, such as 500xCompressor (Li et al., 2025b) and Coconut (Hao et al., 2024), push this further by performing reasoning directly in the latent space. However, recent studies (Li et al., 2025a) show that implicit compression methods tend to have positional bias. This means they often ignore information from the beginning or middle of the context, focusing instead on the most noticeable content and overlooking less prominent details. Also, these implicit methods (Cheng et al., 2025; Wei et al., 2025) tend to lack flexibility, as they often require specially designed post-training processes or the use of latent vectors as new inputs. This limits the applicability of such techniques to advanced API-based models.

Unlike these methods, our design allows our method to preserve global structure while capturing fine-grained details, and crucially, since it operates without latent representations, it ensures seamless compatibility with API-based models.

B.2 Hallucinations in LLMs

Hallucination in LLMs remains a pervasive issue, characterized by the generation of non-factual or unfaithful content (Huang et al., 2024; Si et al., 2025c). Recent research has shifted from simply viewing hallucination as a generation error to a more nuanced perspective of "controllable generation." A substantial body of work has taxonomized the causes of hallucination (Si et al., 2025a; Liu et al., 2025; Ji et al., 2023; Zhao et al., 2025), treating hallucination mitigation as a debiasing task and striving to eliminate LLM hallucinations. However, strictly eliminating such uncertainty may compromise model creativity and usability. (Jiang et al., 2024) propose an alternative view, regarding hallucination as a manifestation of creativity that requires control rather than simply eliminating it. In this way, to mitigate the hallucination, many works attempt to introduce external knowledge integration to establish controllable context for the generation process of LLMs, e.g., RAG technologies. However, these methods (Zhang and Zhang, 2025; Fan et al., 2024) often rely on embedding-based

retrieval stage and struggle with noisy retrieval or context integration due to lack of the structural relations of the retrieved context. Different from these studies, we attempt to establish context for controllable generation through structured context compression. Thus, our structured context compression via the proposed EDU-based Context Compressor can preserve global foresight while capturing fine-grained details, ensuring less noisy information and reducing the hallucination on downstream tasks.

C Method Details

To provide a concrete understanding of our pipeline, we visualize the transformation process from linear text to a structured hierarchy in Figure 3.

C.1 Budget-Aware Coarse-to-Fine Tree Selection

Eq. (5) describes a budget-aware greedy selection. Here we specify our concrete tree-aware implementation, which (i) prioritizes selecting coherent semantic blocks at the same hierarchy level, and (ii) refines them into lower-level (down to leaf) nodes when the remaining budget is insufficient.

Preliminaries. Each node is $n_j = (h_j, l_j, \sigma_j)$ with $\sigma_j = [\texttt{id}_{\text{start}}, \texttt{id}_{\text{end}}]$. The node relevance score s_j is computed by Eq. (4). Let $\mathcal{V}_l = \{n_j \in \mathcal{T} \mid l_j = l\}$ denote the set of nodes at depth l. We use $\text{Child}(n)$ to denote the children of node n.

Span de-duplication via coverage set. To avoid selecting overlapping spans from an ancestor and its descendants, we maintain a EDU index set

$$\mathcal{I}_{\text{cov}} = \bigcup_{n \in \mathcal{C}} \{\texttt{id} \mid \texttt{id} \in \sigma_n\}. \tag{6}$$

A node n is considered *eligible* if it contains at least one uncovered EDU:

$$\text{Eligible}(n) \Leftrightarrow \exists \texttt{id} \in \sigma_n \ \text{ s.t. } \ \texttt{id} \notin \mathcal{I}_{\text{cov}}. \tag{7}$$

Level-first greedy selection. We iterate from coarse levels to fine levels. At each level l, we sort nodes in $\mathcal{V}_l$ by s_j and greedily add an eligible node n to $\mathcal{C}$ if the retrieved length of its *uncovered* span fits the remaining budget:

$$\text{cost}(n) = \text{Len}\big(\text{Retrieve}(\sigma_n) \setminus \mathcal{I}_{\text{cov}}\big). \tag{8}$$

We add n if $\text{cost}(n) \leq B_{\text{rem}}$, where B_{rem} is the remaining token budget.

Budget-triggered refinement (descending to leaves). If a high-scoring internal node n is not affordable, we refine it by descending to $\text{Child}(n)$ and applying the same greedy criterion to its children, where each child is scored by Eq. (4) using its own (h_j, t_{rep}). This refinement is applied recursively until (i) some descendants fit the remaining budget, (ii) leaf nodes are reached, or (iii) the budget is exhausted. This yields a coarse-to-fine allocation of the budget: the algorithm prefers intact high-level sub-trees when possible, and backs off to finer-grained leaf evidence otherwise.

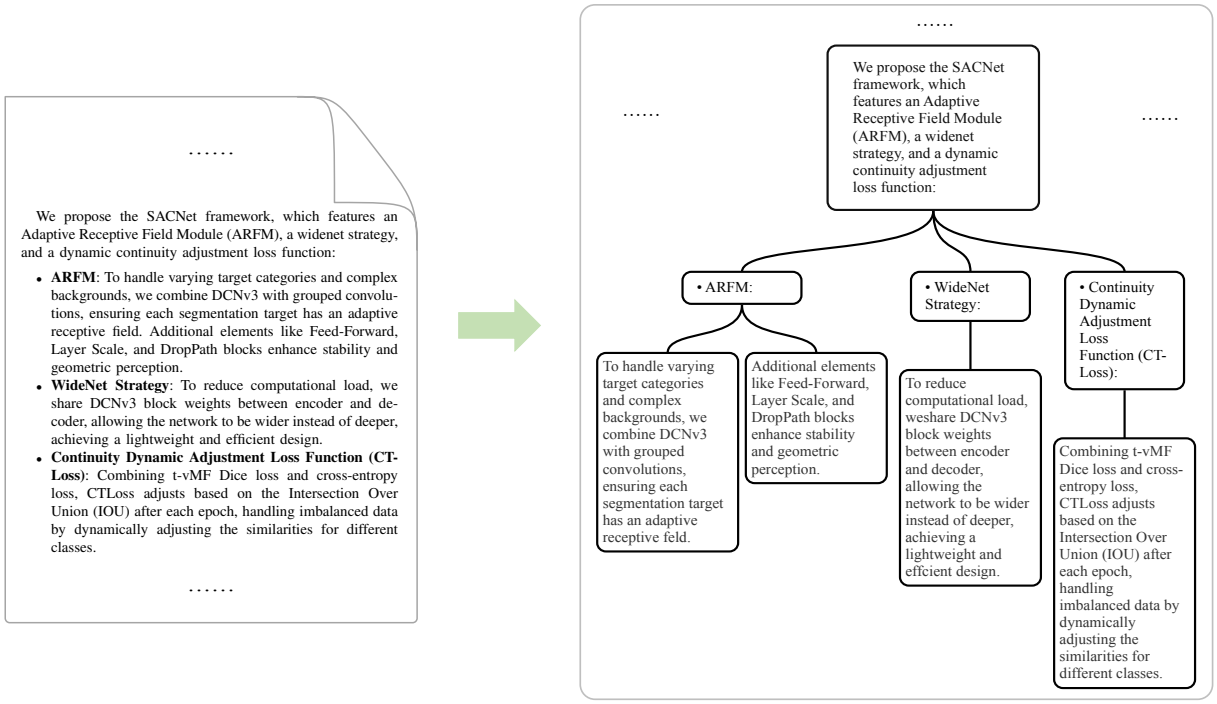
D Dataset Details

D.1 StructBench

We introduce **StructBench**, a manually curated benchmark designed to evaluate document structure understanding and structure-aware context compression in realistic long-document settings. StructBench contains 248 multi-format documents in **English and Chinese**, covering a broad range of document organizations, writing styles, and formatting conventions. The benchmark is constructed from both web-native content (e.g., public articles and content platforms) and PDF-based documents (e.g., academic papers and institutional reports), reflecting the heterogeneous formats commonly encountered in real-world long-context applications.

To improve diversity and reduce domain-specific bias, we collect documents from 10 major domains:

- **Academic Papers/Journals:** Research articles with explicit section hierarchies and formal discourse organization.
- **Government Documents:** Official files, policy mandates, and public announcements with rigid structural templates.
- **Institutional/Business Reports:** Industry analysis, financial reports, and white papers containing nested sections, tables, and summary-style writing.
- **Technical Blogs:** In-depth articles on software, engineering, or science, often mixing headings, lists, code-related explanations, and narrative discussion.
- **Tutorials & Guides:** Step-by-step instructional content with procedural structure.



(a) The document **(b) The structured document tree**

Figure 3: **Qualitative example of Traceable Context Compression. (a) Input Document.** An example of a raw input document. **(b) The Structured Document Tree.** The model outputs a hierarchical index that reorganizes the linear EDUs into semantic clusters (Sections and Subsections). Note that the leaf nodes contain explicit coordinate pointers, allowing the system to faithfully retrieve the exact original text without hallucination.

- **News:** Journalistic reports that combine headline-style summaries with event-centered body paragraphs.
- **Opinion/Analysis:** Editorials, reviews, and commentary articles with relatively loose but rhetorically rich discourse structures.
- **Popular Science:** Educational articles for general audiences that balance explanation, examples, and high-level organization.
- **Books:** Selected excerpts from structured long-form writing.
- **Lifestyle & Entertainment:** Semi-structured content such as travel and arts, which broadens the stylistic coverage of the benchmark.

The validation and test sets are sampled according to the source distribution of the document repository to preserve realistic format and domain diversity. Both sets are fully manually annotated. The training corpus is constructed from the same repository and annotated through a solver–verifier pipeline, whose annotation quality is evaluated on the manually labeled validation set.

The annotation process involves **10 annotators**. Annotation guideline development and annotator training are conducted on the validation set before large-scale annotation begins. The guideline standardizes two key aspects of discourse structure annotation: (1) the hierarchy depth, where annotation proceeds until a *clear discourse structure* is reached, and (2) the segmentation criterion under such clear structures. In practice, the latter introduces relatively limited ambiguity because the target units are structurally explicit.

To ensure annotation consistency, the annotation workflow adopts multiple quality-control stages, including first-pass annotation, second-pass review, expert quality inspection, and model-based spot checking. Inter-annotator agreement is high (**Cohen's** $\kappa > 0.8$), indicating that the target discourse structures are consistently identifiable under the proposed guideline.

This domain design allows StructBench to cover both highly regular documents (e.g., academic papers and government documents) and relatively less constrained writing (e.g., opinion pieces and lifestyle content), making it suitable for evaluating whether a model can generalize across heteroge-

neous discourse structures rather than overfitting to a single document template. The benchmark therefore provides a challenging testbed for structure prediction under substantial variation in layout cues, hierarchy depth, and discourse style.

D.2 General Long-Context Understanding Domain

To evaluate the generalization capabilities of our model beyond structural extraction, we conduct experiments on a diverse suite of general long-context tasks from the LongBench. This evaluation encompasses three distinct categories: Multi-Document QA, Summarization, and Few-shot Learning. The selected datasets cover a wide range of sources, including Wikipedia, government reports, and meeting transcripts and support both English and Chinese languages. As detailed in Table 7, the context lengths vary significantly, ranging from approximately 2k to over 22k tokens, providing a rigorous testbed for assessing robustness in processing extensive unstructured contexts.

We exclude Single-Document QA due to its typically shorter context lengths that do not necessitate information compression, and Code Completion to maintain a focus on natural language scenarios. Furthermore, Synthetic Tasks are omitted as they largely rely on retrieval-based matching (e.g., mapping summaries to source texts) which exhibits a weak correlation with the semantic compression capabilities we aim to assess.

D.3 DeepSearch Domain

HLE: We utilize *Humanity's Last Exam* (HLE) (Phan et al., 2025), an expert-curated benchmark designed to assess frontier-level academic competence. From the original set of 2,500 highly challenging questions spanning multiple disciplines, we focus on the subset of 2,154 text-only questions to evaluate deep reasoning capabilities.

BrowseComp-ZH: To test robustness in a messy information environment, we employ *BrowseComp-ZH* (Zhou et al., 2025). This is the first high-difficulty benchmark evaluating real-world web browsing and reasoning within the Chinese information ecosystem. It comprises 289 complex multi-hop queries across 11 domains (e.g., Film & TV, Technology, Medicine) often embedded in noisy web layouts.

E Experiment Details

E.1 Train Details

We present the detailed training configuration in Table 8. The model is fine-tuned based on the Qwen3-4B architecture. To accommodate the long-context requirement, we extend the maximum sequence length to 32,768 tokens and adjust the Rotary Positional Embedding (RoPE) base frequency to $1,000,000$. We utilize the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon =$ 1e-8. A weight decay of 0.1 and a gradient clipping threshold of 1.0 are applied to stabilize training. The learning rate follows a cosine decay schedule, starting from a peak of 1e-5 and decaying to a minimum of 1e-6, with a linear warmup phase covering the first 10% of the training steps.

E.2 Detailed Model Specifications

To ensure the reproducibility of our experiments, Table 9 lists the specific version identifiers and access paths for all models used in the *Structure Extraction* and *Downstream Long-Context* tasks.

E.3 Robustness Analysis Across Context Lengths

To evaluate the model's capability in maintaining structural constraints over extended inputs, we partition the 248 StructBench documents into 10 bins based on token length. We employ Tree Edit Distance (TED) as the primary metric, where lower scores indicate higher structural fidelity.

As depicted in Figure 4, all models maintain low error rates in short contexts (Avg $<$ 3k tokens). However, distinct performance characteristics emerge as the context length extends. The LingoEDU-4B model demonstrates remarkable resilience, effectively bridging the gap between lightweight and large-scale architectures.

Notably, in the medium-to-long range (e.g., Bin 7, Avg $\sim$4.2k tokens), LingoEDU-4B achieves a TED of 4.68, outperforming both the 1.7B baseline and remarkably the 8B variant (TED 7.52). Furthermore, in the most challenging regime (Bin 9, Avg $>$11k tokens), while the 1.7B model suffers significant degradation (TED 27.44), LingoEDU-4B maintains a competitive performance (TED 23.96), rivaling the stability of the 8B model. This indicates that LingoEDU-4B offers the optimal trade-off, delivering 8B-level long-context robustness with significantly higher inference efficiency.

Dataset	Source	Avg len	Metric	Language	#data
Multi-Document QA					
HotpotQA	Wikipedia	9,151	F1	English	200
2WikiMultihopQA	Wikipedia	4,887	F1	English	200
MuSiQue	Wikipedia	11,214	F1	English	200
DuReader	Baidu Search	15,768	Rouge-L	Chinese	200
Summarization					
GovReport	Government report	8,734	Rouge-L	English	200
QMSum	Meeting	10,614	Rouge-L	English	200
MultiNews	News	2,113	Rouge-L	English	200
VCSUM	Meeting	15,380	Rouge-L	Chinese	200
Few-shot Learning					
TREC	Web question	5,177	Accuracy (CLS)	English	200
TriviaQA	Wikipedia, Web	8,209	F1	English	200
SAMSum	Dialogue	6,258	Rouge-L	English	200
LSHT	News	22,337	Accuracy (CLS)	Chinese	200

Table 7: An overview of the dataset statistics in LongBench used for evaluation. 'Source' denotes the origin of the context. 'Avg len' (average length) is computed using the number of words for English datasets and characters for Chinese datasets. 'Accuracy (CLS)' refers to classification accuracy.

Hyperparameter	Value
Model Configuration	
Base Model	Qwen3-4B
Max Sequence Length	32,768
RoPE Base	1,000,000
Precision	bf16
Optimization	
Optimizer	AdamW
Optimizer Params	$\beta_1 = 0.9, \beta_2 = 0.95$
Peak Learning Rate	1×10^{-5}
Min Learning Rate	1×10^{-6}
LR Scheduler	Cosine
Warmup Ratio	0.1
Weight Decay	0.1
Gradient Clipping	0.5
Batching & Parallelism	
Global Batch Size	128
Training Iterations	1,296
Tensor Parallelism (TP)	4
Sequence Parallelism	True

Table 8: Hyperparameters and configuration used for training the LingEDU.

E.4 Cost Analysis

To evaluate the economic efficiency of our proposed method, we conducted a comprehensive cost comparison between the pure LLM-based pipeline (Baseline) and our LingoEDU-integrated pipeline.

Pricing Model Assumptions. The cost calculations are based on the pricing of *GPT-4.1* . The pricing scheme is **$2.00 per 1M input tokens** and **$8.00 per 1M output tokens**. For the baseline, the LLM handles the entire specific parsing and answering workflow. For our method, the parsing is offloaded to the local LingoEDU-4B model. While local inference is not free due to hardware amortization and electricity, it is significantly cheaper. Based on our deployment statistics, the cost for processing a single document with LingoEDU is approximately **$0.0007**, compared to **$0.0168** with GPT-4.1. This ∼24x cost efficiency allows LingoEDU to process massive amounts of tokens locally with minimal economic impact.

Cost Breakdown. Table 10 details the token consumption and estimated expenses.

Analysis. As shown in Table 10, our strategy achieves the lowest total cost ($5.76), representing a **51.7% reduction** compared to the Direct LLM approach ($11.92) and a **74.7% reduction** compared to the LLM-based Pipeline ($22.74). The data highlights two critical economic advantages:

1. **Avoiding the "Generation Tax" in Parsing:** The LLM Pipeline incurs prohibitively high costs during the Parsing Phase ($22.43) because it relies on GPT-4.1 to generate structured outputs. Generating 1.3M output tokens triggers the expensive prediction rate ($8/M). In contrast, **Ours (LingoEDU)** offloads this heavy structural extraction to local models. This allows us to process the same 5.9M raw tokens for virtually zero cost ($0.53 overhead), completely bypassing commercial API fees for the most data-intensive stage.

2. **Strategic Context Allocation:** Compared to "Direct LLM," which feeds all 5.9M raw to-

Model Name in Paper	Experiment Scope	Real API ID / Checkpoint / Address
OpenAI Models		
GPT-4o	Structure	`gpt-4o-2024-11-20`
GPT-4.1	Structure	`gpt-4.1-2025-04-14`
OpenAI o3	Structure	`o3-2025-04-16`
OpenAI o4-mini	Structure	`o4-mini-2025-04-16`
GPT-5	**Downstream**	`gpt-5-2025-08-07`
Anthropic Models		
Claude-3.7-Sonnet	Structure	`claude-3-7-sonnet-20250219`
Claude-4	Structure	`claude-sonnet-4-20250514`
Claude Opus 4.1	**Downstream**	`claude-opus-4-1-20250805`
Google Models		
Gemini-2.5-flash	Structure	`gemini-2.5-flash`
Gemini-2.5-pro	Structure	`gemini-2.5-pro`
Gemini 3 Pro	**Downstream**	`gemini-3-pro-preview`
DeepSeek Models		
DeepSeek-V3	Structure	`deepseek-v3-250324`
DeepSeek-R1	Structure, **Downstream**	`deepseek-r1-250528`
DeepSeek-V3.1	**Downstream**	`deepseek-v3-1-250821`
DeepSeek-V3.2	**Downstream**	`deepseek-v3-2-251201`
Qwen Models (Local / Open Weights)		
Qwen3-32B	Structure	`https://huggingface.co/Qwen/Qwen3-32B`
Qwen3-235B	Structure, **Downstream**	`https://huggingface.co/Qwen/Qwen3-235B-A22B`
Specialized Tools		
Jina-Reader	Structure	`https://jina.ai/reader`
Firecrawl	Structure	`https://www.firecrawl.dev`

Table 9: Detailed version tracking for all models. "Structure" denotes models used in Table 1 (StructBench), while "Downstream" refers to the Long-Context evaluation tasks. Specific identifiers or URLs are provided in the third column to specify the exact model artifacts used.

kens into the costly API ($11.92), our method uses the local Qwen3 reranker to filter noise efficiently. This reduces the final input volume to 2.6M tokens, cutting input costs by half while maintaining high information density. Conversely, compared to the "LLM Pipeline" (which aggressively reduces context to 148k tokens to save money), we reinvest the savings from the parsing phase into a much richer context (2.6M tokens) for the Answer Phase. This strikes an optimal balance: providing significantly more context than the baseline pipeline to ensure accuracy, while remaining far cheaper than direct processing.

E.5 Impact of Retrieval Granularity on Compression and Performance

To determine the optimal granularity for our structure-aware retrieval, we conduct an ablation study on LongBench by varying the number of retrieved nodes k (e.g., $k \in \{3, 10, 20\}$). We analyze the trade-off between task performance (Metric Score) and computational efficiency (Compression Rate), with the standard Top-100 or Top-200 retrieval serving as the baseline.

As illustrated in Figure 5, increasing k naturally improves performance by incorporating more context; however, it reduces the compression rate, leading to higher computational overhead.

Justification for Choosing $k = 10$. Our results identify $k = 10$ as the optimal operating point.

- **High Fidelity:** At $k = 10$, the model performs comparably to, and in some datasets (e.g., HotpotQA, Musique) even surpasses, the Standard baseline. This indicates that the top-10 identified nodes capture the majority of the task-relevant signal effectively suppressing noise found in the larger top-100 context.
- **Efficiency Gain:** While further increasing k to 20 yields only marginal performance gains (diminishing returns), $k = 10$ maintains a significantly higher compression rate (preserving $> 85\%$ compression on average).

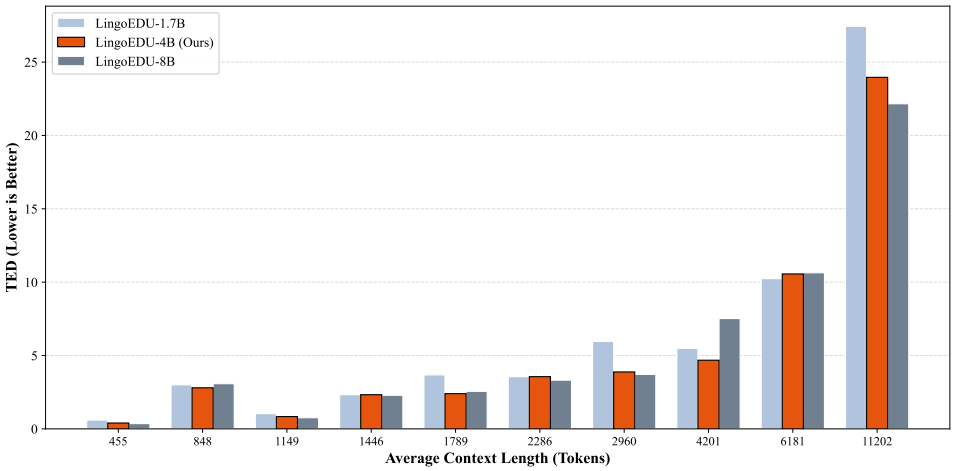


Figure 4: Performance comparison (TED) on StructBench across varying context lengths. The dataset is divided into 10 intervals by token count. Lower TED indicates better performance. The LingoEDU-4B model (orange) demonstrates superior efficiency-robustness balance. It consistently outperforms the 1.7B model in long contexts and achieves comparable—or even superior (e.g., Bin 7)—structural consistency relative to the larger 8B model, validating its selection as the primary backbone.

Consequently, we adopt $k = 10$ as the default setting for our method, as it strikes the most favorable balance between maximizing structural accuracy and minimizing token consumption.

F Prompt Template

F.1 LLM Baselines on StructBench

To strictly pinpoint the structural understanding capabilities of general-purpose Large Language Models (LLMs), we utilized a unified zero-shot system prompt. This prompt essentially instructs the model to act as a parser, extracting the hierarchy without modifying the content. The specific prompt content is visually presented in Figure 6.

It is important to clarify the distinct inference paradigms used in our experiments:

- **LLM-based Baselines (Applied):** This prompt was applied to all models listed in Table E.2, including GPT series, Claude series, DeepSeek series, and Qwen series.
- **Commercial APIs (Not Applied):** Services like *Jina Reader* and *Firecrawl* operate as specialized black-box parsers. They ingest URLs or files and return structure via internal logic, rendering external prompting inapplicable.
- **EDU (Ours) (Not Applied):** Unlike general-purpose LLMs that require detailed instructions (Prompt Engineering) to define the task, our model is explicitly trained via Supervised Fine-Tuning (SFT) for this specific objective. It accepts the raw document stream and outputs the structured tree end-to-end, relying on its internal parametric knowledge rather than prompt-based context.

F.2 LLM Baselines on LongBench

To assess the effectiveness of our proposed framework on LongBench, specifically for the entry labeled **Ours (LingoEDU)** in Table 4, we implemented a two-stage retrieval-augmented generation pipeline. This pipeline leverages LingoEDU for structural parsing and a ranking model for context selection. The specific prompts for both stages are visually presented in Figure 7.

It is crucial to understand how the components interact in our experiment:

- **LingoEDU (Structure Parsing):** First, the raw long context is processed by LingoEDU, which segments the text into a hierarchical tree of Elementary Discourse Units (EDUs). This assigns a unique index ID and a depth level to every meaningful span of text.
- **Ranking Model (Context Selection):** For the QA stage, we do not feed the entire document to the LLM. Instead, a lightweight ranking model scores the relevance of each EDU against the user query. We select the top-k relevant nodes to construct the `{ctxt}` variable, ensuring adherence to the token budget while maintaining high information density.

Stage	Metric	Direct LLM	LLM Pipeline	Ours (LingoEDU)
1. Parsing Phase				
	Method	-	GPT-4.1 Generation	LingoEDU (Local)
	Input Tokens	-	5,955,972	5,955,972
	Output Tokens	-	1,314,406	2,170,766
	Est. Cost	**-**	**$22.43**	**$0.53**
2. Reranking Phase				
	Method	-	Qwen3-0.6B	Qwen3-0.6B
	Tokens	-	1,013,704	2,170,766
	Est. Cost	**-**	**<$0.01**	**<$0.01**
3. Answering Phase				
	Method	GPT-4.1	GPT-4.1	GPT-4.1
	Input Tokens	5,955,972	147,995	2,605,437
	Output Tokens	1,357	1,157	1,475
	Est. Cost	**$11.92**	**$0.31**	**$5.22**
Total	**Total Cost**	**$11.92**	**$22.74**	**$5.76**
	Cost Comparison	**+107%**	**+295%**	**Base**

Pricing: GPT-4.1 ($2.00/1M Input, $8.00/1M Output). Local Reranker (Qwen3-0.6B) cost is negligible (<$0.002).
1: Direct LLM processes raw tokens directly. **2**: LLM Pipeline is expensive due to generating 1.3M tokens during parsing.

Table 10: Cost comparison across three strategies on the **NarrativeQA dataset**. **Direct LLM** incurs high input costs by reading the full raw text. The **LLM Pipeline** attempts to parse data first but becomes the most expensive option due to high API output costs in the parsing stage. **Ours (LingoEDU)** achieves the lowest cost by offloading parsing and reranking to efficient local models, reducing the total cost by **51.7%** compared to Direct LLM and **74.7%** compared to the LLM Pipeline.

- **LLM (Reasoning & Synthesis):** The general-purpose LLM acts as the final reasoner. Crucially, it is instructed to cite the specific node indices (e.g., `[12]`) provided by LingoEDU, allowing for traceable answers grounded in the retrieved segments.

F.3 LLM Baselines on DeepSearch

To handle complex queries requiring multi-step reasoning and verification, specifically on DeepSearch Benchmark, we employ a hierarchical prompting strategy. This strategy coordinates two distinct agent roles: the **Solver** (which generates candidate solutions using tools) and the **Selector** (which verifies and chooses the best solution). Additionally, we implement a **Search Enhancement** module that injects structured retrieval results directly into the reasoning process. The prompt designs for these components are detailed below.

- **Figure 8:** Shows the prompts for the Solver, enabling code execution and web interactions.
- **Figure 9:** Shows the prompts for the Selector, enforcing strict verification protocols.
- **Figure 10:** Illustrates how raw search results are processed via EDU parsing and LLM summarization before injection.

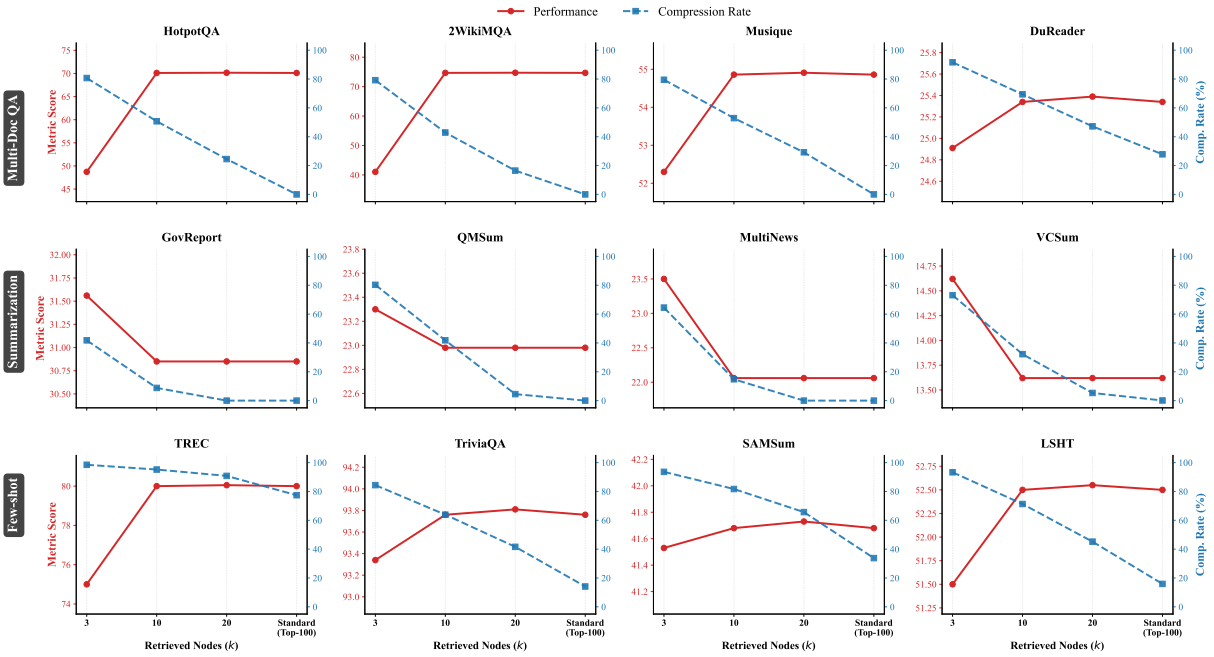


Figure 5: Ablation analysis of retrieval size (k) versus performance and compression on LongBench tasks. The left y-axis denotes the metric score, while the right y-axis shows the input compression rate. The dashed line represents the Standard (Top-100) baseline. We observe that $k = 10$ represents the optimal trade-off: it achieves performance nearly identical to or better than the dense baseline while maintaining a significantly higher compression rate. Increasing k further to 20 provides negligible metric improvements but incurs a steeper cost in context length.

System Prompt used for LLM Baselines on StructBench

Instruction:
Work your way down through the article's heading structure, outputting each level of heading in Markdown format.

- First, use only the original headings—do not include body text, do not summarize, and do not rewrite.

- If a heading is split into multiple sentences due to punctuation, combine them into a single complete heading and output it as one entry.

- Output the heading structure top-down, carefully identifying hierarchical relationships and determining whether overly detailed levels are necessary to output.

- Finally, return only the final result without any additional explanations.

Article: [INPUT CONTENT]

Figure 6: Unified system prompt used for zero-shot baseline evaluation.

Stage 1: Content Summarization Prompt (Indexing Phase)

System Message:
You are a professional content analyst. Please always output valid JSON. **User Prompt:**
Please generate a professional retrieval content based on the following:

- Source: `{source_desc}`
- Title: `{title}` (or 'No explicit title detected')
- Hierarchical Content:
  `{content_text}` *(Note: Formatted with indentation strings matching EDU levels)*

Summarization Requirements:
1) Provide a 150-250 word summary.
2) List 3-5 key points.
3) Outline the main purpose/function.
4) Briefly describe content structure characteristics. **Output JSON Format:**

```
{ "summary": "...", "key_points": ["..."], "main_purpose": "...", "content_structure": "...",
"information_value": "High/Medium/Low" }
```

Stage 2: Retrieval-Augmented QA Prompt (Inference Phase)

System Message:
You are a rigorous retrieval QA assistant. **User Prompt:**
You are a rigorous retrieval QA assistant. Answer only based on the provided context. Do not fabricate information.
Question:
`{query}` **Context (indexed by node ID):**
`{ctxt}`
(Format: '[0] Text... [5] Text...' — selected by the Ranking Model) **Please provide:**

- Direct answer (if derivable).
- Concise explanation (based on the context).
- Citations of the node indices used (e.g., `[12, 15]`).

Requirements:
- If the context is insufficient, explicitly state "Insufficient to answer".
- Do not introduce information outside the provided context.

Figure 7: Unified prompts used for the **Ours (LingoEDU)** pipeline in LongBench. Stage 1 summarizes the structured EDU tree, while Stage 2 performs citation-aware QA using ranked EDU nodes.

(a) Solver User Template (Input Interface)

The problem is: {query} Solve the problem with the help of feedback from a code executor. Every time you write a piece of code between <code> and </code>, the code inside will be executed. [...] Based on the reasoning process and the executor feedback, you could write code to help answering the question for multiple times. **Available Functions:**
- web_search(keywords): Calls a search engine; returns string results. Useful for knowledge questions.
- web_parse(link, query): Parses a specific link for detailed answers.

Constraints:
- Do not be overconfident.
- Put code in <code> snippets.
- Put final answer in <answer> tags with \boxed.

(b) Solver Assistant Prefix (Few-Shot Guidance)

<think>
Okay, to answer the user's question, I will answer user's problem by deep reasoning together with writing python code in <code></code> format. For example: 1. To search: <code>keywords=... results=web_search(keywords) print(results)</code>
2. To parse: <code>link=... results=web_parse(link, query) print(results)</code>
3. To compute: <code>a=123 b=456 print(a+b)</code> Now, let me analyze the user's question.

Figure 8: Prompts used for the **Solver Agent**. The user template (a) defines the tool-use environment, while the assistant prefix (b) primes the model for Chain-of-Thought reasoning paired with Python code execution.

(a) Selector User Template (Verification Interface)

You are a diligent and precise judge. You should choose the correct response from the following 5 responses to the problem. **Your Task:** Verify responses by writing codes. Do not trust information easily. Do not be influenced by majority voting. **Tools:** web_search(query), web_parse(link, query). **Format Requirement:**
- VERIFICATION: [Detailed process for each response]
- CONCLUSION: [Brief summary]
- FINAL DECISION: <select>Response X</select>

Problem: {query}
Response 1: {solution_1}
...
Response 5: {solution_5}

(b) Selector Assistant Prefix (Verification Logic)

<think>Okay, to choose the most correct response... I should verify these responses by writing codes and analyze whether each response is correct. [...Examples of tool usage for verification...] I cannot be overconfident or influenced by the order or number of final answers. Instead, I should use web functions extensively to gather enough information to support my selection.

Figure 9: Prompts used for the **Selector Agent**. This stage employs a "Judge" persona that critically evaluates five candidate solutions using independent tool calls before making a final selection.

Intelligent Summary Generation based on EDU Parsing

System: You are a professional search result analyst... always return valid JSON. **User Prompt:**
Please generate a professional summary for the search result based on the hierarchical content structure:

- **Query:** `{query}`
- **URL / Title:** `{url}` / `{title}`
- **Hierarchical Content:** `{main_content}` (from EDU parsing)
- **Extracted Key Points:** `{key_points}`

Requirements: 1. Analyze relevance to query. 2. Concise summary (100-200 words). 3. Highlight relevant info. 4. Identify 3-5 key points. 5. Evaluate credibility. **Output JSON:** `{ "summary": "...", "key_points": ["..."], "relevance_score": "...", "content_quality": "...", "main_topics": ["..."] }`

Figure 10: Search Enhancement Prompt. When EDU parsing is enabled, this prompt converts raw hierarchical web content into a structured, relevance-scored JSON summary, which is then injected into the Solver's context.