RETHINKING RETRIEVAL IN THE LONG-CONTEXT ERA: TRAINING-FREE SEMANTIC TREES FOR ADAPTIVE CONTEXT CONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-context language models can increasingly process entire documents, making full-context prompting a practical alternative to passage retrieval. However, greater capacity determines how much information a model can receive, but not which information best supports a query. This distinction motivates rethinking retrieval: when the full document already fits, can selecting and organizing its evidence improve answering? The challenge is to retain supporting facts and relations while keeping the context focused on the question. To reconcile these goals, we introduce TREEFUSE, a training-free framework that uses semantic abstraction to guide selection while retaining original text for answering. TREEFUSE constructs a reusable semantic hierarchy from the full document in a single generation, linking node descriptions to source spans. Given a query, structural–semantic scoring and fine-grained lexical attribution nominate shared source nodes; their union with ancestor context preserves structural relations and complete local text. Across four frontier answer models, TREEFUSE improves average QA-F1 by 1.72 points over full-context prompting on three LongBench tasks, surpassing the evaluated baselines on the four-model average while reducing answer-context length by 58.55%. Our code is available at this anonymous link.

1 INTRODUCTION

Frontier language models can increasingly accommodate entire documents within their context windows, making full-context prompting a practical option for long-document question answering. Yet input capacity and context selection address different questions: how much information a model can receive, and which information it should receive for a particular query. Prior work shows that performance can depend strongly on where relevant information appears within a long context, even when that information remains available to the model (Liu et al., 2024). Prior compression methods show that selecting context can improve answering as well as shorten the input (Jiang et al., 2024a; Chirkova et al., 2025), while LaRA (Li et al., 2025) finds no universally preferable choice between retrieval-based and full-context approaches. These findings motivate rethinking retrieval beyond accommodating input-length constraints: **When the full document already fits within the context window, can retrieval improve answering, and how should it select the supporting context?**

A relevant passage is not always sufficient to support an answer. A question may require a directly mentioned fact together with background or a relation established elsewhere in the document. Sufficient Context (Joren et al., 2025) explicitly distinguishes contextual support from answer correctness. Context construction must therefore consider both the support it retains and how selectively it presents that support. Narrow extraction can omit a necessary fact or separate a detail from its interpretation, whereas broad expansion may introduce unrelated material. Controlled studies of arithmetic reasoning show that such irrelevant information can interfere with model predictions (Shi et al., 2023), motivating attention to context concentration as well as support retention. Hierarchical methods such as HiChunk (Lu et al., 2026) and LongRefiner (Jin et al., 2025) make document organization available to retrieval and refinement. The remaining design question is how to use that organization while remaining sensitive to precise source-level details.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

To reconcile support retention with context concentration, we separate the information used to select evidence from the text supplied for answering. Figure 1 illustrates the goal: retaining supporting facts and their relation while excluding unrelated material. We instantiate this principle with TREEFUSE, a training-free framework that uses semantic abstraction to guide selection while preserving original evidence. **Step 1: Source-grounded Semantic Tree Construction.** A pre-trained model jointly proposes semantic boundaries and hierarchical relations from the full document in a single generation, linking node descriptions to original source spans. The resulting hierarchy is query-independent and reusable. **Step 2: Query-adaptive Context Construction.** Structural–semantic scoring and fine-grained lexical attribution nominate nodes from this shared representation. A localized lexical match can retain a complete source node, supplementing semantic judgments with precise details. The selected nodes and their ancestors form a pruned hierarchy that preserves the original parent–child relations and complete local text. The same constructed context can be supplied to different answer models without model-specific fine-tuning.

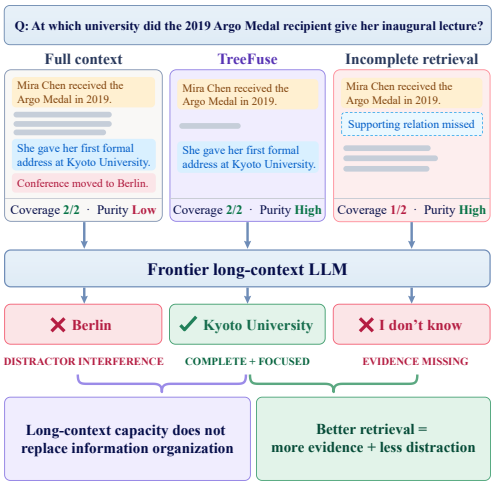

Figure 1: A schematic example contrasts full context with distractors, incomplete retrieval with missing support, and TREEFUSE's focused context retaining both supporting statements.

We evaluate TREEFUSE with four frontier answer models that offer strong long-context processing capabilities: Qwen3.8-Max (Qwen Team, 2026b), DeepSeek-V4-Pro (DeepSeek AI, 2026), Kimi-K2.6 (Kimi, 2026), and GLM-5.3 (GLM-5-Team et al., 2026). Full-context prompting provides a direct reference for testing whether retrieval remains beneficial when the complete document is already available to the answer model. Across three LongBench QA tasks, TREEFUSE reduces the answer-context length by 58.55% on average while improving the four-model average by 1.72 QA-F1 points over full-context prompting. The gains are more pronounced on the multi-hop HotpotQA and 2WikiMQA tasks, where the four-model averages improve by 1.72 and 2.46 points, respectively. On 2WikiMQA with Kimi-K2.6, TREEFUSE achieves a 7.00-point improvement, corresponding to a 14.76% relative gain. Evaluation on LaRA further tests transfer across heterogeneous long-document tasks, where TREEFUSE achieves the highest four-model average accuracy among the evaluated methods with 43.87% less answer context. Beyond answer scores, source-evidence analyses examine completeness and concentration, while component comparisons characterize semantic and lexical selection, hierarchical organization, and single-pass construction. These evaluations address both parts of our research question, establishing average gains over full-context prompting and examining how the selected contexts retain and organize supporting evidence.

To summarize, our contributions are:

- We empirically demonstrate that query-adaptive context construction can improve average answering performance even when full-document input is feasible, positioning retrieval as a complement to long-context capacity.
- We introduce reusable semantic trees and dual-granularity selection to coordinate structural support with precise source details, retaining complete local text without method-specific training.
- We provide a broad evaluation across four frontier answer-model families and diverse long-document tasks, obtaining a 1.72-point average QA-F1 improvement on the three LongBench tasks with 58.55% less context. We release our code to support research on adaptive context construction: `https://anonymous.4open.science/r/treefuse-ECFE`.

2 RELATED WORK

Retrieval and Compression for Long-Context Question Answering LongBench (Bai et al., 2024) and LaRA (Li et al., 2025) evaluate long-context understanding and retrieval-based answering. LongRAG by Jiang et al. (2024b) groups related documents into larger retrieval units to exploit

long-context models. A separate LongRAG by Zhao et al. (2024b) jointly addresses global information and factual details in long-context QA. Compression methods act at different granularities: LongLLMLingua (Jiang et al., 2024a) prunes tokens, Provence (Chirkova et al., 2025) selects sentences, and RECOMP (Xu et al., 2024) and BRIEF-Pro (Gu et al., 2026) learn or generate compact contexts. ACC-RAG (Guo et al., 2025) adapts compression rates, while SARA (Jin et al., 2026) combines textual and compressed representations. Sufficient Context (Joren et al., 2025) examines whether supplied information is sufficient for answering.

Semantic Chunking and Hierarchical Document Indexing Semantic chunkers such as LumberChunker (Duarte et al., 2024) and Meta-Chunking (Zhao et al., 2024a) seek coherent boundaries, with libraries such as Chonkie[1] providing multiple implementations. Hierarchical indexes additionally organize relationships among units. RAPTOR (Sarthi et al., 2024) recursively clusters and summarizes passages. TreeRAG (Tao et al., 2025) combines tree-based chunking with bidirectional traversal. SiReRAG (Zhang et al., 2025b) constructs similarity and relatedness trees and combines their nodes in a retrieval pool. Heading-aware indexing (Tinh & Phuong, 2025) exploits existing sections, while HiChunk (Lu et al., 2026) and LongRefiner (Jin et al., 2025) learn specialized components for hierarchical chunking or refinement.

Structure-Aware and Hybrid Retrieval BM25 (Robertson & Zaragoza, 2009) and DPR (Karpukhin et al., 2020) provide lexical and learned semantic relevance signals. Contextual Retrieval (Anthropic, 2024) prepends explanatory context to chunks before embedding and BM25 indexing. MemWalker (Chen et al., 2023) interactively navigates summary trees, while PageIndex (Zhang et al., 2025a) uses reasoning-based search over document hierarchies. HiChunk (Lu et al., 2026) expands retrieval hits through Auto-Merge, and LongRefiner (Jin et al., 2025) performs query-aware refinement. DISRetrieval (Chen et al., 2026) combines discourse parsing, LLM-enhanced node representations, and structure-guided hierarchical retrieval.

TREEFUSE approaches retrieval as a complement to long-context capacity, not solely a remedy for limited input length. It targets answer contexts that retain supporting evidence while concentrating on the query, using a reusable semantic hierarchy to guide source-preserving selection. Appendix A.1 provides a detailed comparison with evaluated baselines and selected related methods.

3 METHOD

3.1 PROBLEM FORMULATION AND OVERVIEW

Given a document D and query q, TREEFUSE constructs an answer context $C(q, D)$ by selecting source-grounded semantic units. The objective is to retain supporting information while limiting content unrelated to the query, even when the full document fits within the model's context window.

Source selection does not create new source evidence; it changes which content is visible to the answer model and how that content is organized. TREEFUSE therefore separates the information used to select evidence from the text delivered for answering. A compact semantic view exposes a node's topic and structural role, while its original text retains details not captured by the description. Selecting complete local units preserves the surroundings of a match, and ancestor completion retains structural connections without including every neighboring branch.

TREEFUSE combines a reusable hierarchy with query-dependent selection and pruning. Figure 2 outlines the pipeline. First, source-grounded semantic tree construction links node descriptions and hierarchical relations to original source spans (§3.2). Given a query, structural–semantic scoring and fine-grained lexical attribution nominate nodes from this shared representation (§3.3). Their union is completed with ancestor nodes and used to prune the original tree, yielding an answer context that preserves retained structural relations and complete local source text (§3.4).

3.2 SOURCE-GROUNDED SEMANTIC TREE CONSTRUCTION

The document-to-hierarchy mapping in Figure 2 links document structure to original source text. A pretrained language model receives the full document and jointly proposes semantic boundaries and

[1] https://docs.chonkie.ai/

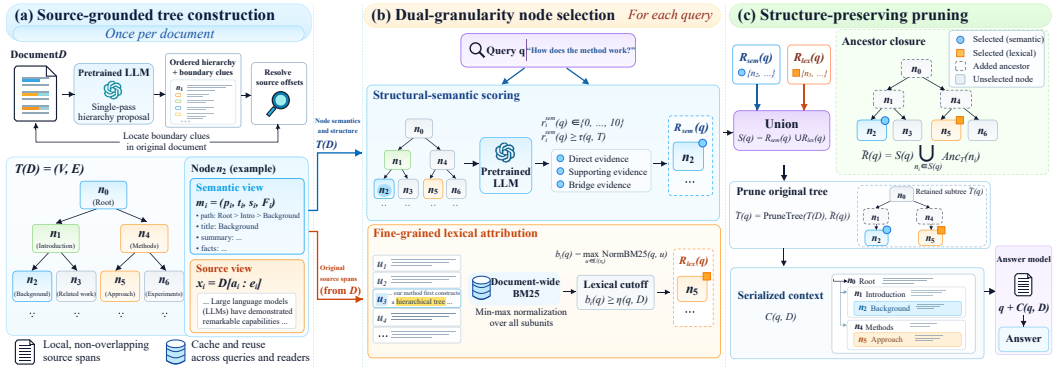

Figure 2: Overview of TreeFuse. (a) A single generation constructs a semantic hierarchy whose nodes are aligned to original source spans. (b) Structural–semantic scoring and fine-grained lexical attribution independently nominate nodes from this shared representation. (c) Their union is completed with ancestor nodes and used to prune the original tree. Retained nodes preserve their structural relations and complete local source text for answering.

parent–child relations in one main generation. Its output contains an ordered hierarchy, node titles, summaries, atomic facts, and verbatim boundary clues. A source-alignment procedure resolves these clues to document offsets, yielding a hierarchy $T(D) = (V, E)$.

Each node $n_i \in V$ has a semantic view m_i and a source view x_i:

$$m_i = (p_i, t_i, s_i, F_i), \qquad x_i = D[a_i : e_i], \tag{1}$$

where p_i is its structural path, t_i its title, s_i its summary, F_i its atomic facts, and $[a_i, e_i)$ its source interval. Each node owns a local source span rather than the concatenation of its descendants. Its identity links the semantic view, source interval, and position in the hierarchy, so semantic scoring and lexical matching can nominate the same underlying source unit. The descriptions provide an indexing view; the original spans provide answer evidence. This source-grounded hierarchy is constructed independently of queries and can be reused across queries and answer models. Construction prompts, source alignment, and a case illustrating the tree's utility for downstream QA appear in Appendices A.2, A.3, and E.5, respectively.

3.3 Query-Adaptive Dual-Granularity Selection

The two channels in the dual-granularity selection module of Figure 2 nominate nodes from the shared hierarchy using complementary views. Structural–semantic scoring assesses a node's relevance and supporting role from its descriptions and structural context. Fine-grained lexical attribution identifies localized matches in its original text, including entities, values, or phrases not captured by the descriptions. The channels produce node sets independently rather than averaging scores with different scales and meanings. Their selections may overlap: a node can be nominated by either channel or both.

Structural–semantic scoring. For each query, a pretrained model rates every node on a 0–10 relevance scale using its semantic view, available parent metadata, and document-level structural context. The prompt requests judgments of direct relevance and supporting relationships. All nodes are scored, so a child remains eligible even if its parent scores low. The initial selection is

$$R^0_{\text{sem}}(q) = \{n_i \in V : r^{\text{sem}}_i(q) \geq \tau(q, T)\},$$

where $r^{\text{sem}}_i(q)$ is the node's relevance rating and $\tau(q, T)$ the semantic cutoff. A nonempty-selection safeguard yields the final set $R_{\text{sem}}(q)$. Appendix A.3 details the cutoff policy and safeguards.

Fine-grained lexical attribution. Each source span x_i is divided into sentence- or paragraph-like subunits $\mathcal{U}(n_i)$. We build a shared BM25 index over the document-wide pool $\mathcal{U} = \bigcup_i \mathcal{U}(n_i)$, min–max normalize scores across this pool, and attribute the strongest match to its enclosing node:

$$b_i(q) = \max_{u \in \mathcal{U}(n_i)} \text{NormBM25}(q, u), \qquad R_{\text{lex}}(q) = \{n_i \in V : b_i(q) \geq \eta(q, D)\}. \tag{2}$$

A node without subunits receives a zero lexical score. A localized match can therefore nominate a complete node without requiring its entire source span to have high lexical overlap. The lexical attribution channel in Figure 2 maps fine-grained matches to their enclosing source nodes: the subunit determines the matching granularity, while the node determines the retention granularity. The lexical cutoff $\eta(q, D)$ follows the P90 rule in Appendix A.3, which also specifies score normalization.

3.4 Structure-Preserving Pruning and Context Construction

The union and ancestor-closure step in Figure 2 determines the node set for a query-specific pruned tree. We first take the union of the two candidate sets:

$$S(q) = R_{\text{sem}}(q) \cup R_{\text{lex}}(q). \tag{3}$$

Lexical matching supplements semantic selection rather than filtering it, so a node nominated by either channel remains eligible for retention. We then add the ancestors of the selected nodes:

$$\widetilde{R}(q) = S(q) \cup \bigcup_{n_i \in S(q)} \text{Anc}_T(n_i). \tag{4}$$

Ancestor completion preserves the paths connecting retained nodes within the original hierarchy. It does not expand a selected node to every surrounding branch: each added ancestor contributes only its own local source span, not the text of all its descendants. We prune the original tree to this ancestor-closed node set:

$$\widetilde{T}(q) = \text{PruneTree}\big(T(D), \widetilde{R}(q)\big). \tag{5}$$

The resulting tree preserves parent–child relations among retained nodes while removing unselected branches. The answer context $C(q, D)$ is formed from their complete local source text and structural information. Semantic summaries and atomic facts guide selection but do not substitute for the source evidence. Benchmark-specific input formats are detailed in Appendix A.4.

Construction and selection require no method-specific parameter updates. The hierarchy is reusable across queries, while each query induces its own selected subtree. For a fixed query, the constructed context can be shared across answer models. Algorithm 1 in Appendix A.4 provides the complete selection and pruning procedure.

4 Experiments

4.1 Experimental Setup

Implementation. For *source-grounded tree construction (§3.2)*, we use Qwen3.6-35B-A3B (Qwen Team, 2026a) to generate the semantic hierarchy and node descriptions, and a forward source matcher to align generated boundary clues with original document spans. The resulting trees are cached for reuse. For *dual-granularity node selection (§3.3)*, the semantic channel uses the same Qwen3.6-35B-A3B model to score nodes against the query, while the lexical channel uses node-level BM25 scores obtained by taking the maximum normalized score over each node's source-text subunits. For *structure-preserving pruning (§3.4)*, we take the union of the two selections, add ancestor nodes, and prune unselected branches while preserving parent–child links and complete local source text. LongBench receives the resulting tree-structured JSON; LaRA formats the retained nodes as numbered source chunks with ancestor-title provenance. We evaluate these contexts with Qwen3.8-Max (Qwen Team, 2026b), DeepSeek-V4-Pro (DeepSeek AI, 2026), Kimi-K2.6 (Kimi, 2026), and GLM-5.3 (GLM-5-Team et al., 2026), reusing each query's constructed context across answer models without model-specific fine-tuning and holding it fixed across three answer-generation runs. Construction and selection prompts, selection settings, and input formats are documented in Appendices A.2, A.3, and A.4.

Benchmarks and metrics. We evaluate context construction across complementary QA settings and document domains, with four frontier answer models testing the reuse of the same selected contexts. Following prior compression research (Jiang et al., 2024a), LongBench (Bai et al., 2024) provides scientific-document QA through Qasper (Dasigi et al., 2021) and multi-passage reasoning

Table 1: LongBench QA-F1 (three-run means). Overall averages four models; parentheses show changes from Raw. Bold/underline mark the best/second-best means.

Dataset	Method	Qwen3.8-Max	DeepSeek-V4-Pro	Kimi-K2.6	GLM-5.3	Overall
Qasper	Raw	48.89	<u>51.20</u>	<u>50.35</u>	46.61	<u>49.26</u>
	LongLL (Jiang et al., 2024a)	29.75	26.55	24.58	27.72	27.15 (↓22.11)
	Provence (Chirkova et al., 2025)	47.50	46.18	48.17	46.84	47.17 (↓2.09)
	HiChunk (Lu et al., 2026)	49.70	49.88	49.24	46.85	48.92 (↓0.35)
	BRIEF-Pro (Gu et al., 2026)	<u>49.93</u>	48.50	47.98	<u>47.06</u>	48.37 (↓0.90)
	LongRefiner (Jin et al., 2025)	31.52	31.00	29.91	29.59	30.50 (↓18.76)
	TREEFUSE (ours)	**50.96**	**51.35**	**50.62**	**48.05**	**50.25** (↑0.98)
HotpotQA	Raw	<u>75.69</u>	<u>72.20</u>	59.47	**70.68**	<u>69.51</u>
	LongLL (Jiang et al., 2024a)	67.69	55.48	49.11	53.92	56.55 (↓12.96)
	Provence (Chirkova et al., 2025)	71.44	66.21	58.05	68.35	66.01 (↓3.50)
	HiChunk (Lu et al., 2026)	70.53	69.57	58.16	68.21	66.62 (↓2.89)
	BRIEF-Pro (Gu et al., 2026)	73.80	69.61	<u>60.22</u>	69.09	68.18 (↓1.33)
	LongRefiner (Jin et al., 2025)	64.64	53.24	54.89	57.80	57.64 (↓11.87)
	TREEFUSE (ours)	**77.39**	**73.64**	**63.50**	<u>70.38</u>	**71.23** (↑1.72)
2WikiMQA	Raw	<u>84.19</u>	<u>79.28</u>	47.44	<u>71.62</u>	<u>70.63</u>
	LongLL (Jiang et al., 2024a)	75.87	59.16	35.61	55.53	56.54 (↓14.09)
	Provence (Chirkova et al., 2025)	78.87	55.32	46.15	66.97	61.82 (↓8.81)
	HiChunk (Lu et al., 2026)	80.65	**79.54**	47.75	70.79	69.69 (↓0.95)
	BRIEF-Pro (Gu et al., 2026)	81.70	70.22	<u>54.27</u>	71.17	69.34 (↓1.29)
	LongRefiner (Jin et al., 2025)	65.53	32.50	38.57	46.59	45.80 (↓24.84)
	TREEFUSE (ours)	**84.85**	79.00	**54.44**	**74.08**	**73.09** (↑2.46)

through HotpotQA (Yang et al., 2018) and 2WikiMultihopQA (Ho et al., 2020). We follow its official answer templates and normalized token-overlap QA-F1 scoring. LaRA-32K (Li et al., 2025) complements these tasks with 887 questions spanning 64 books, scientific papers, and financial reports, testing transfer across document domains. We report accuracy using LaRA's official task-specific grading prompts with a fixed gpt-oss-20b judge. All methods and answer models share the same question set per dataset across three runs. Scores average the runs; Overall equally averages the four model scores on this common set. The LongBench summary weights its three tasks equally. Context reduction compares mean answer-context character counts with Raw on aligned examples. Sample alignment is detailed in Appendix B.1.

Baselines. **Raw context** passes the uncompressed context and question to the answer model. Baselines include **LongLLMLingua** (Jiang et al., 2024a) (**LongLL**) for query-aware token compression; **Provence** (Chirkova et al., 2025) for sentence pruning; **HiChunk** (Lu et al., 2026) for hierarchical chunking and retrieval; **BRIEF-Pro** (Gu et al., 2026) for generative compression; and adapted **LongRefiner** (Jin et al., 2025) for query-aware document refinement. We use reported construction settings where applicable. To compare performance at similar mean context lengths, we also adjust baseline budgets or pruning thresholds toward TREEFUSE's retained length, reporting achieved lengths and scores. See Appendices B.3 and C.3 for details.

4.2 MAIN RESULTS

Tables 1 and 2 summarize the LongBench and LaRA comparisons under the evaluation protocol in §4.1. We examine answering performance across tasks and answer models, followed by an analysis of the evidence retained in the constructed contexts. Run-level confidence intervals are reported in Appendix C.1. Compression–performance analysis and comparisons at adjusted context lengths are provided in Appendices C.2 and C.3.

Higher average QA-F1 with shorter answer contexts. TREEFUSE achieves the highest four-model average QA-F1 among the evaluated methods on each of the three LongBench tasks. Across

tasks, it improves over Raw by **1.72 QA-F1 points while reducing answer-context length by 58.55%**. The gains are larger on the two multi-hop tasks: TREEFUSE reaches Overall scores of 71.23 on HotpotQA and 73.09 on 2WikiMQA, exceeding Raw by 1.72 and 2.46 points, respectively. On 2WikiMQA with Kimi-K2.6, the score rises from 47.44 to 54.44—a *7.00-point absolute improvement and a 14.76% relative gain*. The larger gains on tasks requiring information integration across passages are consistent with the motivation for structure-conditioned selection. These results directly address our research question: even when the complete document is available to the answer model, selecting and organizing its content can improve average answering performance rather than merely reduce input length. Appendix C.4 quantifies when answer-input savings offset amortized tree construction and query selection under model-weighted input-token accounting.

Gains across answer models with shared contexts. TREEFUSE improves the LongBench average for **every evaluated answer model**, despite all four backbones already offering strong long-context capabilities. The largest gain occurs on Kimi-K2.6, whose average rises from 52.42 to 56.19—an absolute improvement of 3.77 points and a relative gain of 7.19%. Qwen3.8-Max, DeepSeek-V4-Pro, and GLM-5.3 improve by 1.48, 0.44, and 1.20 points, respectively. This consistency across model families suggests that the method is not tied to a particular backbone. These gains require *no answer-model-specific context construction or fine-tuning*: the same prepared contexts are reused across models. They show that access to the full document does not eliminate the value of upstream information organization and selection.

Cross-benchmark performance. LaRA extends the evaluation to heterogeneous document domains and question types. Under the main-comparison settings in Table 2, all evaluated external baselines exhibit substantial declines in four-model average accuracy relative to Raw, ranging from 6.21 to 39.27 points. In contrast, TREEFUSE achieves the **highest four-model average accuracy with 43.87% less answer context**, increasing accuracy from 80.17 to 80.59. DeepSeek-V4-Pro gains 1.62 accuracy points over Raw. This contrast shows that substantial context reduction need not come at the expense of average answer quality in this setting. Together with the LongBench results, it supports the applicability of source-grounded context construction across different long-document evaluation settings.

Table 2: LaRA accuracy (%) averaged over three answer runs. Overall averages the four models, with changes from Raw in parentheses.

Method	Qwen	DS	Kimi	GLM	Overall
Raw	79.26	76.52	80.66	**84.22**	80.17
LongLL	64.45	61.14	62.30	64.93	63.20 (↓16.96)
Provence	71.85	70.77	71.93	76.96	72.88 (↓7.29)
HiChunk	72.15	72.23	73.85	77.59	73.96 (↓6.21)
BRIEF-Pro	72.30	71.14	70.73	75.31	72.37 (↓7.79)
LongRefiner	43.07	38.79	38.68	43.04	40.89 (↓39.27)
TREEFUSE	**79.37**	**78.14**	**80.89**	83.94	**80.59** (↑0.42)

Evidence completeness and concentration. Table 3 relates annotated evidence preservation to answer quality; the *Overall* column directly reuses the four-model QA-F1 values from Table 1. All-hit measures complete coverage under the evidence-matching rule, Evidence Purity measures the concentration of selected content on annotated source evidence, and ECP is their equally weighted mean. The evidence metrics use annotated subsets, with definitions and matching procedures provided in Appendices D.1–D.3. TREEFUSE achieves the highest Overall on all three datasets, together with the **highest ECP on Qasper (52.95) and 2WikiMQA (62.01)** and the second-highest on HotpotQA (42.11). The component scores clarify these ECP results. On Qasper, TREEFUSE preserves nearly the same complete-evidence coverage as HiChunk (96.57% versus 96.00%) while achieving higher purity (9.33% versus 5.80%). On HotpotQA, it exceeds HiChunk on both measures. BRIEF-Pro's higher

Table 3: Evidence coverage, purity, and ECP (%). Overall QA-F1 is reused from Table 1.

Dataset	Method	All-hit	Purity	ECP	Overall
Qasp.	LongLL	0.57	2.18	1.38	27.15
	Prov.	16.57	17.82	17.20	47.17
	HiChunk	96.00	5.80	50.90	48.92
	BRIEF	73.71	**25.41**	49.56	48.37
	LongRef.	20.57	5.19	12.88	30.50
	TREEFUSE	**96.57**	9.33	**52.95**	**50.25**
Hotpot	LongLL	0.00	28.95	14.48	56.55
	Prov.	0.00	47.67	23.84	66.01
	HiChunk	12.23	52.42	32.33	66.62
	BRIEF	0.00	**92.61**	**46.31**	68.18
	LongRef.	0.00	49.00	24.50	57.64
	TREEFUSE	**12.77**	71.44	42.11	**71.23**
2Wiki	LongLL	2.55	35.53	19.04	56.54
	Prov.	0.00	52.81	26.41	61.82
	HiChunk	**55.10**	43.28	49.19	69.69
	BRIEF	8.16	**97.93**	53.05	69.34
	LongRef.	0.51	34.81	17.66	45.80
	TREEFUSE	41.33	82.69	**62.01**	**73.09**

purity gives it the highest ECP on this task, whereas TREEFUSE obtains higher Overall QA-F1 (71.23 versus 68.18). On 2WikiMQA, TREEFUSE combines the second-highest All-hit and purity to attain the highest ECP. These comparisons characterize a combination of support retention and concentration, rather than dominance in either component alone. This pattern is consistent with the design of selecting structurally relevant source nodes and supplementing them with precise lexical matches while preserving the local context of retained evidence.

4.3 ABLATION STUDIES

We examine how semantic and lexical selection complement each other, what semantic units and hierarchy contribute, and how tree construction affects answering. Tables 4–6 compare variants on HotpotQA and 2WikiMQA, both requiring cross-passage evidence integration. Qwen3.8-Max answers questions; Qwen3.6-35B-A3B constructs and scores nodes. Scores average three runs. Avg. is the unweighted two-task mean, with changes from Raw in parentheses. BM approximately matches mean retained characters, not per-question token counts. Best means are bold and runners-up underlined. See Appendix E.1 for implementation details.

Semantic and lexical selection. We test *whether the two selection channels contribute beyond increasing context length* by comparing three variants in Table 4. *Semantic-only* retains semantic selections and their ancestor context, removing lexical supplementation. *Lexical-only (BM)* uses lexical node selection with ancestor closure under an approximately matched character budget. *Semantic-only (BM)* instead supplements semantic selections with random nodes using a single fixed seed, approaching the full method's context length without targeted lexical matching. TREEFUSE reaches **81.12 average QA-F1**, exceeding Semantic-only by 0.84 points, Lexical-only (BM) by 6.16 points, and Semantic-only (BM) by 2.72 points. The smaller difference from Semantic-only indicates that semantic selection accounts for most of the observed performance, with lexical matching providing an incremental contribution. On HotpotQA and 2WikiMQA, the full method exceeds Semantic-only by 0.81 and 0.88 points, whereas random supplementation scores 1.66 and 2.10 points lower than Semantic-only. These contrasts favor **targeted supplementation over the tested random-expansion configuration**, rather than attributing the improvement simply to additional context.

Table 4: Effects of semantic and lexical selection.

Variant	HotpotQA	2WikiMQA	Avg.
Raw	75.69	<u>84.19</u>	79.94
Semantic-only	<u>76.59</u>	83.97	<u>80.28</u> (↑0.34)
Lexical-only (BM)	73.54	76.38	74.96 (↓4.98)
Semantic-only (BM)	74.93	81.87	78.40 (↓1.54)
TREEFUSE	**77.39**	**84.85**	**81.12** (↑1.18)

To examine whether these performance differences are accompanied by differences in evidence retention, we match each variant's saved context against the mapped gold annotations used in our source-evidence analysis (Appendix D.1). Span Recall measures length-weighted coverage of annotated source evidence. Table 16 in Appendix E.3 reports the conditional results. Its pooled column shows that, on the annotated subsets across all three tasks, including Qasper, TREEFUSE has 21.50 percentage points higher Span Recall and 4.45 points higher QA-F1 than Lexical-only (BM), pooling over questions. Relative to the original Semantic-only configuration, the corresponding differences are 1.17 percentage points and 0.36 QA-F1. These contrasts are consistent with structural–semantic selection providing most evidence coverage and lexical matching serving as a smaller, targeted supplement. The HotpotQA and 2WikiMQA columns of Table 16 also show that the full method obtains 2.42 and 2.76 points higher QA-F1 than Semantic-only (BM) on their annotated subsets, despite small differences in aggregate Span Recall. This pattern emphasizes *which source details are retained*, rather than treating additional context as interchangeable.

Semantic units and hierarchy. We examine *what semantic units and hierarchical organization contribute* through two types of comparisons in Table 5. First, we compare TREEFUSE with TREEFUSE *(flat)*, which retains the same source-node inventory but removes the hierarchy. This comparison evaluates the joint use of hierarchy-aware scoring, ancestor completion, and structured serialization. The selected con-

Table 5: Effects of semantic units and hierarchy.

Variant	HotpotQA	2WikiMQA	Avg.
Raw	<u>75.69</u>	<u>84.19</u>	<u>79.94</u>
Fixed 200	72.77	79.47	76.12 (↓3.82)
Chonkie	74.33	78.77	76.55 (↓3.39)
TREEFUSE (flat)	74.07	80.93	77.50 (↓2.44)
TREEFUSE	**77.39**	**84.85**	**81.12** (↑1.18)

tent and retained length can also change, so it is not a same-context test of structure alone. We additionally compare *Fixed 200* and *Chonkie*, which replace the semantic units with fixed 200-token chunks and Chonkie semantic chunks, respectively. These representation-level comparisons change unit boundaries and metadata, complementing the same-inventory hierarchy comparison. The hierarchy-enabled pipeline achieves **81.12 average QA-F1 versus 77.50 for the flat pipeline**, a 3.62-point difference across the two multi-hop tasks. The differences are 3.32 points on HotpotQA and 3.92 points on 2WikiMQA. Fixed 200 and Chonkie yield average scores of 76.12 and 76.55, falling below the full method by 5.00 and 4.57 points, respectively. The same-inventory comparison is consistent with hierarchical processing being useful for selecting support distributed across passages, while the alternative-unit comparisons support the use of the source-grounded semantic representation within this selection framework.

Tree-construction strategies. We ask *how joint tree construction compares with staged and adapted representations*. Table 6 compares these strategies under the same downstream selection framework. The *Multi-stage* configuration uses the same Qwen3.6-35B-A3B model but separates construction into three stages: extracting and extending a document outline from grouped source chunks; validating section boundaries and anchoring them to the original text; and enriching the resulting nodes with summaries and facts. The *HiChunk adapter* maps HiChunk-derived units into the node representation used by our selector, rather than evaluating the complete HiChunk retrieval system. Implementation and configuration details are provided in Appendix E.2. Within the tested selector, Qwen3.8-Max answer model, and two multi-hop tasks, the single-pass constructor achieves **81.12 average QA-F1**, compared with 78.90 for Multi-stage and 78.34 for the HiChunk adapter. Its advantage over Multi-stage is 2.22 points on average, with differences of 1.92 points on HotpotQA and 2.52 points on 2WikiMQA. These results demonstrate that the tested single-generation constructor can provide an effective representation for this selection pipeline. Together with the hierarchy comparison in Table 5, indicating that using document structure effectively does not require a more elaborate construction procedure.

Table 6: Effects of tree-construction strategy.

Variant	HotpotQA	2WikiMQA	Avg.
Raw	75.69	84.19	79.94
Multi-stage	75.47	82.33	78.90 (↓1.04)
HiChunk adapter	74.15	82.54	78.34 (↓1.60)
TREEFUSE	**77.39**	**84.85**	**81.12** (↑1.18)

5 CONCLUSION

We revisited retrieval in the long-context era as a problem of selecting and organizing evidence, rather than merely fitting documents into a limited window. TREEFUSE operationalizes this view through a training-free, reusable semantic hierarchy: abstraction guides selection, while structure-preserving pruning retains original text for answering. Across four frontier answer models, TREEFUSE improves average QA-F1 by 1.72 points over full-context prompting on three LongBench tasks with 58.55% less answer context, and achieves the highest four-model average accuracy on LaRA with 43.87% less context. Under the main evaluation settings, all compared retrieval and compression baselines fall below Raw in four-model average performance on each LongBench task and LaRA, whereas TREEFUSE exceeds it. This contrast validates the value of adaptive context construction while showing that input reduction alone does not ensure better answers. The central design principle is to use abstraction to decide what to retain without substituting it for the source evidence. Long-context capacity and retrieval thus address complementary decisions: how much information can be made available, and which source content and relations should be presented for a particular question. Our findings establish the latter as a consequential design choice even when the former no longer prevents full-document input.

Limitation and Future Work. Our evaluation centers on long-document question answering. Extending TREEFUSE to multimodal documents would require linking textual evidence with tables and figures. Although retained text is source-grounded, selection still depends on generated structure and descriptions. Controlled perturbation studies could quantify this sensitivity and guide more robust tree construction and selection. Future work could also explore other evidence-intensive tasks and incremental tree updates as documents evolve, extending the reusable hierarchy beyond the settings evaluated here.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing, specifically for language polishing, grammar correction, and improving the clarity of exposition in text that was first drafted by the authors. We also used generative AI tools as coding assistants for routine, non-novel implementation tasks (e.g., boilerplate code, plotting scripts, and refactoring). We have not used generative AI tools to generate research ideas, formulate the core hypotheses, design the method, produce or analyze experimental results, or draw scientific conclusions; these were carried out entirely by the authors.

ETHICS STATEMENT

TREEFUSE uses public benchmarks and involves no human-subject studies or collection of new personal data. Dataset and model use follows their respective terms. Source-preserving selection does not guarantee factual correctness or privacy protection; sensitive applications require appropriate safeguards.

REPRODUCIBILITY STATEMENT

Our code is available through the anonymous repository linked in the paper. The appendices document prompts, selection rules, experimental settings, and evaluation procedures. Fixed contexts are reused across answer models and repeated runs; API updates and stochastic generation may affect exact replication.

REFERENCES

Anthropic. Introducing contextual retrieval. Anthropic Engineering Blog, 2024. URL https://www.anthropic.com/engineering/contextual-retrieval.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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)*, pp. 3119–3137, 2024.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*, 2023.

Huiyao Chen, Yi Yang, Yinghui Li, Meishan Zhang, Baotian Hu, and Min Zhang. Beyond chunking: Discourse-aware hierarchical retrieval for long document question answering. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18176–18198, 2026.

Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. Provence: efficient and robust context pruning for retrieval-augmented generation. In *International Conference on Learning Representations*, volume 2025, pp. 38082–38106, 2025.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 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*, pp. 4599–4610, 2021.

DeepSeek AI. DeepSeek V4 technical documentation. Official technical documentation, April 2026. URL https://fe-static.deepseek.com/chat/transparency/deepseek-V4-model-card-EN.pdf.

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

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, Jiajie Zhang, Jian Jiao, Jiaqi Guo, Jingsen Wang, Jingzhao Du, Jinzhu Wu, Kedong Wang, Lei Li, Lin Fan, Lucen Zhong, Mingdao Liu, Mingming Zhao, Pengfan Du, Qian Dong, Rui Lu, Shuang-Li, Shulin Cao, Song Liu, Ting Jiang, Xiaodong Chen, Xiaohan Zhang, Xuancheng Huang, Xuezhen Dong, Yabo Xu, Yao Wei, Yifan An, Yilin Niu, Yitong Zhu, Yuanhao Wen, Yukuo Cen, Yushi Bai, Zhongpei Qiao, Zihan Wang, Zikang Wang, Zilin Zhu, Ziqiang Liu, Zixuan Li, Bojie Wang, Bosi Wen, Can Huang, Changpeng Cai, Chao Yu, Chen Li, Chengwei Hu, Chenhui Zhang, Dan Zhang, Daoyan Lin, Dayong Yang, Di Wang, Ding Ai, Erle Zhu, Fangzhou Yi, Feiyu Chen, Guohong Wen, Hailong Sun, Haisha Zhao, Haiyi Hu, Hanchen Zhang, Hanrui Liu, Hanyu Zhang, Hao Peng, Hao Tai, Haobo Zhang, He Liu, Hongwei Wang, Hongxi Yan, Hongyu Ge, Huan Liu, Huanpeng Chu, Jia'ni Zhao, Jiachen Wang, Jiajing Zhao, Jiamin Ren, Jiapeng Wang, Jiaxin Zhang, Jiayi Gui, Jiayue Zhao, Jijie Li, Jing An, Jing Li, Jingwei Yuan, Jinhua Du, Jinxin Liu, Junkai Zhi, Junwen Duan, Kaiyue Zhou, Kangjian Wei, Ke Wang, Keyun Luo, Laiqiang Zhang, Leigang Sha, Liang Xu, Lindong Wu, Lintao Ding, Lu Chen, Minghao Li, Nianyi Lin, Pan Ta, Qiang Zou, Rongjun Song, Ruiqi Yang, Shangqing Tu, Shangtong Yang, Shaoxiang Wu, Shengyan Zhang, Shijie Li, Shuang Li, Shuyi Fan, Wei Qin, Wei Tian, Weining Zhang, Wenbo Yu, Wenjie Liang, Xiang Kuang, Xiangmeng Cheng, Xiangyang Li, Xiaoquan Yan, Xiaowei Hu, Xiaoying Ling, Xing Fan, Xingye Xia, Xinyuan Zhang, Xinze Zhang, Xirui Pan, Xu Zou, Xunkai Zhang, Yadi Liu, Yandong Wu, Yanfu Li, Yidong Wang, Yifan Zhu, Yijun Tan, Yilin Zhou, Yiming Pan, Ying Zhang, Yinpei Su, Yipeng Geng, Yong Yan, Yonglin Tan, Yuean Bi, Yuhan Shen, Yuhao Yang, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yurong Wu, Yutao Zhang, Yuxi Duan, Yuxuan Zhang, Zezhen Liu, Zhengtao Jiang, Zhenhe Yan, Zheyu Zhang, Zhixiang Wei, Zhuo Chen, Zhuoer Feng, Zijun Yao, Ziwei Chai, Ziyuan Wang, Zuzhou Zhang, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Jia-Chen Gu, Junyi Zhang, Di Wu, Yuankai Li, Kai-Wei Chang, and Nanyun Peng. Brief-pro: Universal context compression with short-to-long synthesis for fast and accurate multi-hop reasoning. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14221–14241, 2026.

Shuyu Guo, Shuo Zhang, and Zhaochun Ren. Enhancing rag efficiency with adaptive context compression. 2025.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1658–1677, 2024a.

Ziyan Jiang, Xueguang Ma, and Wenhu Chen. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*, 2024b.

Jiajie Jin, Xiaoxi Li, Guanting Dong, Yuyao Zhang, Yutao Zhu, Yongkang Wu, Zhonghua Li, Ye Qi, and Zhicheng Dou. 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)*, pp. 3502–3520, 2025.

Yiqiao Jin, Kartik Sharma, Vineeth Rakesh, Yingtong Dou, Menghai Pan, Mahashweta Das, and Srijan Kumar. Sara: Selective and adaptive retrieval-augmented generation with context compression. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14508–14528, 2026.

Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ankur Taly, and Cyrus Rashtchian. Sufficient context: A new lens on retrieval augmented generation systems. In *International Conference on Learning Representations*, volume 2025, pp. 20310–20334, 2025.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Kimi. Kimi K2.6 Tech Blog: Advancing open-source coding. Kimi Official Technical Blog, 2026. URL https://www.kimi.ai/blog/kimi-k2-6.

Kuan Li, Liwen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Shuai Wang, and Minhao Cheng. Lara: Benchmarking retrieval-augmented generation and long-context llms–no silver bullet for lc or rag routing. 2025.

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

Wensheng Lu, Keyu Chen, Zhifeng Shen, Ruizhi Qiao, and Xing Sun. Hichunk: Evaluating and enhancing retrieval augmented generation with hierarchical chunking. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29738–29753, 2026.

Qwen Team. Qwen3.6-35B-A3B: Agentic coding power, now open to all, April 2026a. URL https://qwen.ai/blog?id=qwen3.6-35b-a3b.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026b. URL https://qwen.ai/blog?id=qwen3.8.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, volume 2024, pp. 32628–32649, 2024.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227, 2023.

Wenyu Tao, Xiaofen Xing, Yirong Chen, Linyi Huang, and Xiangmin Xu. Treerag: Unleashing the power of hierarchical storage for enhanced knowledge retrieval in long documents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 356–371, 2025.

Pham Doan Tinh and Luong Ngoc Phuong. Optimizing context retrieval for rag via heading-aware chunking and hierarchical document structure integration. In *2025 IEEE 7th Symposium on Computers & Informatics (ISCI)*, pp. 456–461. IEEE, 2025.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Mingtian Zhang, Yu Tang, and PageIndex Team. Pageindex: Next-generation vectorless, reasoning-based rag, September 2025a. https://pageindex.ai/blog/pageindex-intro.

Nan Zhang, Prafulla Kumar Choubey, Alexander Fabbri, Gabriel Bernadett-Shapiro, Rui Zhang, Prasenjit Mitra, Caiming Xiong, and Chien-Sheng Wu. Sirerag: Indexing similar and related information for multihop reasoning. In *International Conference on Learning Representations*, volume 2025, pp. 100413–100432, 2025b.

Jihao Zhao, Zhiyuan Ji, Yuchen Feng, Pengnian Qi, Simin Niu, Bo Tang, Feiyu Xiong, and Zhiyu Li. Meta-chunking: Learning text segmentation and semantic completion via logical perception. *arXiv preprint arXiv:2410.12788*, 2024a.

Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. Longrag: A dual-perspective retrieval-augmented generation paradigm for long-context question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22600–22632, 2024b.

APPENDIX CONTENTS

A Method Details and Relation to Prior Work 15
A.1 Comparison with Related Methods 15
A.2 Prompt Templates and Output Schemas 15
A.3 Tree Construction, Scoring, and Selection 16
A.4 Structure-Preserving Serialization 17

B Experimental Configurations and Evaluation 19
B.1 Benchmarks and Sample Alignment 19
B.2 Answer Models and Judge Configurations 19
B.3 Baseline Implementations and Context Budgets 19

C Supplementary Quantitative Results 20
C.1 Main-Comparison Details and Run-Level Intervals 20
C.2 Compression–Performance Trade-off 20
C.3 Comparisons with Adjusted Context Lengths 20
C.4 Context Savings and Construction Amortization 21

D Source-Evidence Analysis 24
D.1 Evidence Annotations and Source Alignment 24
D.2 Evidence Metric Definitions 24
D.3 Equal-Weight ECP Rationale 24

E Ablation Details and Case Studies 25
E.1 Selection, Representation, and Construction Variants 25
E.2 Tree-Construction Configurations 25
E.3 Conditional Evidence and Answer Analysis 26
E.4 Semantic Support and Lexical Supplementation Cases 26
E.5 Semantic Adequacy versus Boundary Exactness 27

Table 7: Comparison with evaluated baselines and selected related methods. 'Reuse' denotes a document representation shared across queries. 'Train' denotes method-specific parameter updates.

Method	Indexed representation	Query-time mechanism	Reader input	Reuse / Train
LongLLMLingua	Flat prompt	Token pruning	Retained tokens	No / No
Provence	Flat passages	Sentence pruning	Source sentences	No / Yes
RAPTOR	Summary tree	Dense node retrieval	Summaries / nodes	Yes / No
LongRefiner	Document hierarchy	Learned adaptive refinement	Refined spans	Yes / Yes
HiChunk	Learned hierarchy	Retrieval + Auto-Merge	Merged chunks	Yes / Yes
ACC-RAG	Multi-granular embeddings	Adaptive compression rate	Compressed features	Yes / Yes
SARA	Text + semantic vectors	Fixed-budget reranking	Snippets + vectors	Yes / Yes
BRIEF-Pro	Flat passages	Generative compression	Synthesized brief	No / Yes
TREEFUSE	**Semantic hierarchy linked to local source spans**	**Structure-conditioned scoring and lexical attribution over shared nodes**	**Pruned hierarchy with complete local source text**	**Yes / No**

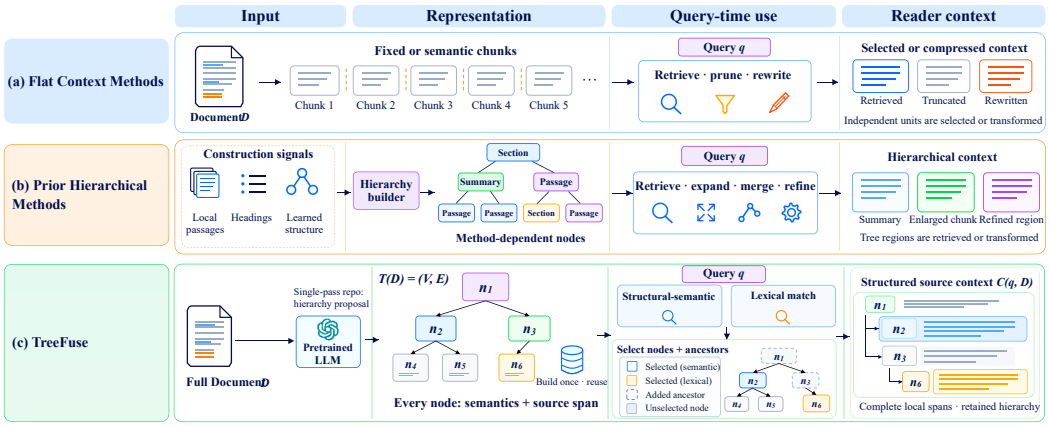

Figure 3: Three paradigms for constructing an answer context. Flat methods organize text as independent units and retrieve, prune, or rewrite them; prior hierarchical methods build method-dependent trees and retrieve, expand, merge, or refine tree regions; TreeFuse constructs a reusable source-grounded semantic hierarchy from the full document, prunes it to an ancestor-closed retained subtree for each query, and serializes the complete local source spans while preserving the original parent–child relations and sibling order.

A METHOD DETAILS AND RELATION TO PRIOR WORK

A.1 COMPARISON WITH RELATED METHODS

Table 7 compares TREEFUSE with evaluated baselines and selected related methods in document representation, query-time selection, and the evidence supplied for answering.

A.2 PROMPT TEMPLATES AND OUTPUT SCHEMAS

The following condensed prompt descriptions summarize the instructions and active interfaces. They are presentation summaries, not verbatim replacements for the templates retained in the supplementary source files.

Tree construction.

Table 8: Principal interfaces used for construction, selection, and answering. Only fields relevant to the all-node selection path are shown.

Interface	Fields
Construction	`sections`: records with `structure`, `title`, `boundary_type`, `heading_verbatim`, `text_before_cut`, `summary`, and `facts`.
Scoring input	Query, document directory, and node records containing `node_id`, `title`, and semantic `material`.
Scoring output	`scores`: records with `node_id`, integer `relevance` in $[0, 10]$, and `reason`.
Selection cutoff	Integer `keep_threshold` in $[1, 10]$.
Answer context	`doc_id` and `tree`, with `level`, `title`, `source_text`, and optional `children` per node.

```
Organize the full document into a hierarchical section tree in one
↪ pass.
Use headings and coherent topic changes as boundaries. Keep passages
separate and sections in document order. Give each section a unique
hierarchical path with its parent present. Copy boundary clues
↪ verbatim.
Summarize only the section's own text and request at most eight atomic,
source-supported facts, preserving exact entities, dates, and numbers.
Return a JSON object containing the sections and required fields.
Document: {document_text}
```

Node relevance.

```
Given the query, document directory, and node metadata, score every
↪ node
from 0 to 10 for its contribution to answering. Use titles, summaries,
atomic facts requested during construction, and available parent
↪ context.
Judge direct evidence, supporting relations, comparisons, temporal
↪ facts,
and cross-node links rather than word overlap alone. Give high scores
↪ to
necessary support even when its wording differs from the question.
Return {"scores":[{"node_id":"...","relevance":0,"reason":"..."}]}.
Query: {query}; Directory: {directory}; Nodes: {nodes}
```

Selection cutoff.

```
Choose a keep threshold from 1 to 10 using the query, scored outline,
score histogram, and retention profile. Preserve plausible supporting
nodes while excluding unrelated branches. Return the numeric cutoff.
Query: {query}; Outline: {tree_outline}; Histogram: {histogram};
Retention profile: {keep_curve}; Node count: {total}
```

A.3 Tree Construction, Scoring, and Selection

Tree construction and source alignment. Qwen3.6-35B-A3B proposes section paths, titles, summaries, atomic facts, and verbatim boundary clues. A forward source matcher resolves boundaries from headings, boundary text, or titles. Consecutive offsets delimit each node's local text, with the first span starting at zero and the last ending at the document boundary. Parent paths define the hierarchy without duplicating child text inside parent nodes.

Scoring inputs. The scorer uses titles, summaries, atomic facts, and available parent context. The construction request limits these facts to eight per section. When both summary and facts are absent,

the scorer uses a short source-text prefix. Breadth-first batches include a document directory and score all nodes before pruning. Table 9 summarizes the implementation settings.

Table 9: Method settings. Entries marked as defaults refer to the implementation.

Setting	Value or behavior	Provenance
Constructor / scorer	Qwen3.6-35B-A3B	Reported setting
Construction facts	Typically 3–6, at most 8 requested	Construction prompt
Scoring material	Section summary, atomic facts, and parent context	Node metadata
Missing metadata	First 300 source characters when both summary and facts are absent	Scorer implementation
Batch size	40 nodes and 48,000 serialized-entry characters	Scorer defaults
Cutoff suggestions	3 attempts, minimum valid keep threshold	Selector default
Coverage floors	2% / 3 nodes, or 5% / 5 nodes for detected location queries	Selector defaults
Lexical cutoff	P90 for all datasets	Selector defaults
Node output	Complete local source text with retained ancestor structure	Serialization interface

Threshold selection and safeguards.

Semantic cutoff. Node ratings lie in $[0, 10]$. The selector requests three cutoff suggestions and uses the minimum valid value in $[1, 10]$. It lowers this cutoff as needed to approach the coverage floor, including ancestor context. Default floors are 2% of source characters and three nodes, increasing to 5% and five nodes for detected location questions. A fallback handles missing cutoff responses, and an empty selection is replaced by the highest-scoring root and its subtree.

Lexical cutoff. BM25 is computed over a shared pool of source-text subunits. Short fragments below 40 characters are merged into their preceding unit when one exists. For nonconstant scores, normalization is $z_u = (b_u - b_{\min})/(b_{\max} - b_{\min})$ over the complete document pool. Node scores are the maxima of their subunit scores, with zero assigned to a node without subunits. For every dataset, the lexical cutoff is the 90th percentile (P90) of the node-score distribution. The percentile uses linear interpolation and includes ties, so it does not select exactly 10% of nodes. Ancestor closure can further enlarge the set.

A.4 Structure-Preserving Serialization

PruneTree retains selected nodes and their ancestors without changing parent–child links or sibling order. LongBench serializes this tree as nested records containing titles, hierarchy labels, and complete local source text. LaRA uses numbered source chunks with ancestor-title provenance. The same selected content is reused across answer models. The benchmark-specific output format is denoted by ρ in Algorithm 1.

Serialized input example. This excerpt from a saved Qasper context preserves its actual field names, titles, and source prefixes. Ellipses omit text for display only. Gold answers and relevance scores are not part of the answer-model input.

```
{"doc_id":"qasper-1-1704f1d50b2554ae","tree":[
  {"level":"level2","title":"Pooling Strategies",
   "source_text":"POOLING STRATEGIES\nIn any language identification
   ↪  model, we want to obtain utterance level representation ...",
   "children":[
     {"level":"level2.1","title":"NetVLAD pooling",
      "source_text":"POOLING STRATEGIES ::: NetVLAD pooling\nThe
      ↪  NetVLAD pooling strategy was initially developed for place
      ↪  recognition ..."}
   ]}
]}
```

Algorithm 1 TREEFUSE query-adaptive tree selection and serialization

Input: cached hierarchy $T(D) = (V, E)$, query q
Settings: cutoff policies $\Pi_{\text{sem}}, \Pi_{\text{lex}}$, serialization rule ρ
$r^{\text{sem}} \leftarrow \text{StructSemScore}(T(D), q)$
$\tau \leftarrow \Pi_{\text{sem}}(q, T(D), r^{\text{sem}})$
$R^0_{\text{sem}} \leftarrow \{n_i \in V : r^{\text{sem}}_i \geq \tau\}$
$R_{\text{sem}} \leftarrow \text{EnsureNonempty}(R^0_{\text{sem}}, T(D), r^{\text{sem}})$
$\mathcal{U}(n_i) \leftarrow \text{Split}(x_i)$ for every $n_i \in V$
$\mathcal{U} \leftarrow \bigcup_{n_i \in V} \mathcal{U}(n_i)$
$z_u \leftarrow \text{NormBM25}(q, u, \mathcal{U})$ for every $u \in \mathcal{U}$
$b_i \leftarrow \max\big(\{z_u : u \in \mathcal{U}(n_i)\} \cup \{0\}\big)$ for every $n_i \in V$
$\eta \leftarrow \Pi_{\text{lex}}(q, D, \{b_i\}_{n_i \in V})$
$R_{\text{lex}} \leftarrow \{n_i \in V : b_i \geq \eta\}$
$S \leftarrow R_{\text{sem}} \cup R_{\text{lex}}$
$\widetilde{R} \leftarrow S \cup \bigcup_{n_i \in S} \text{Anc}_T(n_i)$
$\widetilde{T} \leftarrow \text{PruneTree}(T(D), \widetilde{R})$
Return $C(q, D) \leftarrow \text{SerializeTree}(\widetilde{T}, \rho)$

LongBench prepends the document-content instruction to this JSON. The label `level2.1` denotes a child of `level2`, not a new relevance rank. LaRA instead formats the retained source text as numbered chunks and attaches ancestor-title provenance where available. This format change does not change which nodes were selected.

B Experimental Configurations and Evaluation

B.1 Benchmarks and Sample Alignment

For each dataset, all evaluated methods, all four answer models, and all three answer-generation runs use the same question set. LongBench examples are aligned by question and reference answers, and LaRA examples by task, context type, question, and reference answer. Each method–model score is averaged over the same questions and three runs. Overall is the equally weighted mean of the four model scores, so every method is evaluated on the same question–model combinations.

B.2 Answer Models and Judge Configurations

LongBench conditions use the benchmark's task-specific answer templates. LaRA uses full-document prompts for Raw and retrieved-context prompts for compressed inputs. Frozen context files are reused across answer runs. LaRA uses a fixed gpt-oss-20b judge.

B.3 Baseline Implementations and Context Budgets

Table 10 summarizes the context-construction settings, which are separate from the answer-model configuration. The additional comparisons adjust compression budgets or pruning thresholds toward the retained length of TreeFuse. These targets do not impose identical per-question input sizes.

Table 10: Models and context settings for the main baseline comparison.

Method	Compression or retrieval model	Configuration
LongLL	Llama-2-7B	Requested retention rate 0.25.
Provence	DeBERTa-v3 reranker	Sentence threshold 0.1, title retention enabled, no reordering.
HiChunk	Youtu-HiChunk and BGE-M3	200-token chunks, maximum depth 10, top-128 retrieval and Auto-Merge, 4,096-token context budget.
BRIEF-Pro	BRIEF-Pro checkpoint	20-sentence instruction, 1,024-token output limit.
LongRefiner	Qwen2.5-3B-Instruct	Query analysis, document structuring, and global selection, with BGE reranking and a 2,048-token target.

For the adjusted comparisons, LongLL targets retention rates of 0.60/0.30/0.30 on Qasper/HotpotQA/2WikiMQA and 0.60 on LaRA. Provence uses threshold 0.001 on LongBench and 0.0001 on LaRA. HiChunk changes its context budget, while BRIEF-Pro uses a 240-sentence instruction and a 16,384-token output limit. LongRefiner targets the corresponding mean character ratios of TreeFuse through its adaptation layer. Actual retained lengths are reported in Table 12.

Table 11: Run-level variability for the main comparison. Each entry is the four-model Overall mean with its 95% Student-t interval over three decoding runs. The intervals characterize run variability on the fixed evaluation sample and contexts only.

Dataset	Raw	LongLL	Provence	HiChunk	BRIEF-Pro	LongRefiner	TreeFuse
Qasper	49.26 ±0.99	27.15 ±1.05	47.17 ±0.90	48.92 ±0.48	48.37 ±0.74	30.50 ±0.35	50.25 ±0.22
HotpotQA	69.51 ±0.72	56.55 ±1.08	66.01 ±0.78	66.62 ±2.41	68.18 ±0.51	57.64 ±1.87	71.23 ±1.31
2WikiMQA	70.63 ±0.73	56.54 ±0.87	61.82 ±0.70	69.69 ±0.23	69.34 ±1.16	45.80 ±0.45	73.09 ±0.09
LaRA	80.17 ±0.30	63.20 ±0.29	72.88 ±0.28	73.96 ±1.14	72.37 ±1.19	40.89 ±1.48	80.59 ±0.37

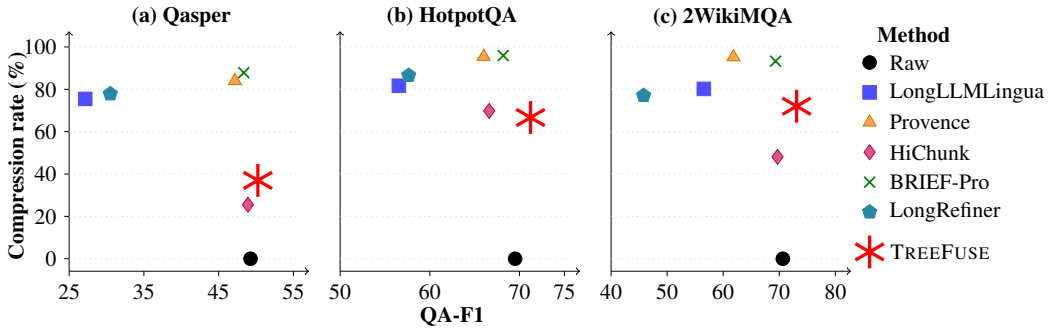

Figure 4: Compression–performance trade-off on Qasper, HotpotQA, and 2WikiMQA, averaged over Qwen3.8-Max, DeepSeek-V4-Pro, Kimi-K2.6, and GLM-5.3. Compression rate is the percentage of raw context characters removed. The vertical axis starts below 0% so Raw remains visible. Moving right indicates better QA-F1, whereas moving upward indicates stronger compression.

C Supplementary Quantitative Results

C.1 Main-Comparison Details and Run-Level Intervals

Tables 1 and 2 report the main comparison. Scores are averaged over three answer runs. Overall first averages models within a task, and the LongBench summary gives equal weight to its three tasks. Table 11 retains the run-level means and intervals from the main comparison.

C.2 Compression–Performance Trade-off

Figure 4 relates four-model average QA-F1 to the proportion of context removed on each LongBench task. We define compression rate as $100\times(1-|C_m|/|C_{\text{raw}}|)$, using mean context character counts on the aligned examples. Points farther right indicate better answering, while points higher in the plot indicate greater compression. TreeFuse achieves the highest average QA-F1 in all three panels while removing 58.55% of the context on average. More aggressive compression, as in BRIEF-Pro, produces shorter inputs but lower average QA-F1. Additional comparisons in Appendix C.3 adjust baseline budgets toward TreeFuse's average retained-context length. TreeFuse retains a positive LongBench-average margin over the strongest external baseline for each answer model in these comparisons. Appendix C.4 extends the analysis to successful-path input-side work, including amortized tree construction and query-selection inputs.

C.3 Comparisons with Adjusted Context Lengths

Table 12 adjusts baseline budgets toward the retained length of TreeFuse and reports the resulting character ratios. These are not exact per-query budget matches. Within these comparisons, all methods and all four answer models use the same question set for each dataset across three answer runs, following the alignment procedure in Appendix B.1. The generation configurations follow Appendix B.2.

Table 12: Additional comparisons with adjusted baseline context lengths. LongBench reports mean QA-F1 across the three tasks and LaRA reports accuracy. Context is the mean percentage of raw-context characters retained across the corresponding models and tasks.

Benchmark	Method	Qwen	DeepSeek	Kimi	GLM	Context (%)
LB	Raw	69.59	67.56	52.42	63.02	100.00
	LongLLMLingua	63.11	55.90	42.69	53.21	38.76
	Provence	69.43	67.32	53.10	63.45	35.96
	HiChunk	66.54	64.62	50.27	61.11	38.96
	BRIEF-Pro	68.71	63.84	54.28	62.29	23.75
	LongRefiner	55.43	41.95	42.02	45.11	30.89
	TREEFUSE	**71.07**	**68.00**	**56.19**	**64.21**	41.46
LaRA	Raw	79.23	76.55	80.70	**84.73**	100.00
	LongLLMLingua	75.09	72.04	74.69	78.85	55.64
	Provence	78.74	78.05	79.87	84.57	47.61
	HiChunk	77.80	76.14	78.93	82.05	51.46
	BRIEF-Pro	76.19	74.33	76.96	80.96	16.94
	LongRefiner	44.92	41.75	42.57	45.31	19.88
	TREEFUSE	**79.35**	**78.17**	**81.01**	84.04	56.13

C.4 CONTEXT SAVINGS AND CONSTRUCTION AMORTIZATION

Reducing the answer context also introduces tree-construction and query-selection work. We quantify this trade-off using the archived successful-path input-token accounting, denoted P1. This model counts the inputs to one successful tree construction per document, one relevance-scoring pass and three cutoff requests per query, and the final answer request. Local BM25 selection adds no LLM request. We use input-side work because context construction directly changes how many tokens each stage must read while holding the question, answer model, answer instruction, decoding configuration, and generation budget fixed. P1 measures whether amortized preprocessing is covered by reduced reader prefill under an explicit reader-to-preprocessor weight.

Preprocessing and reuse. Let D be the number of indexed documents and N the number of associated questions. With $Q = N/D$ questions per document, the mean preprocessing input per question is

$$C_{\mathrm{pre}} = \frac{T_{1,\mathrm{in}}}{N} + \frac{T_{2,\mathrm{in}}}{N} = \frac{T_{1,\mathrm{in}}/D}{Q} + \frac{T_{2,\mathrm{in}}}{N},$$

where $T_{1,\mathrm{in}}$ and $T_{2,\mathrm{in}}$ are the total successful-path construction and selection inputs. Table 13 reports their amortized contributions. LaRA shares 64 document trees across 887 questions, reducing construction input to 2,209 tokens per question despite its longer documents. This reuse affects construction, while query-conditioned selection is charged separately for every question.

Table 13: Preprocessing input-token accounting. Construction and selection columns are per-question Qwen3.6 inputs. D and N describe the indexed collection used to amortize preprocessing. Values are rounded.

Dataset	D	N	N/D	Construction	Selection	C_{pre}
Qasper	148	200	1.351	4,865	14,318	19,183
HotpotQA	200	200	1.000	14,827	33,418	48,245
2WikiMQA	200	200	1.000	8,903	25,403	34,306
LaRA	64	887	13.859	2,209	19,485	21,694

Model-weighted input work. Let T_{raw} and T_{sel} be the answer model's mean input tokens per question with full and selected contexts. We use κ for its relative input-scale weight, distinct from the serialization rule ρ in Appendix A.4 (Algorithm 1). The comparison is

$$C^{\mathrm{P1}}_{\mathrm{raw}} = \kappa T_{\mathrm{raw}}, \qquad C^{\mathrm{P1}}_{\mathrm{ours}} = C_{\mathrm{pre}} + \kappa T_{\mathrm{sel}}.$$

Table 14: Break-even thresholds and modeled reductions in proxy-weighted successful-path input work under the stated κ. Raw and TREEFUSE are Qwen3.6 input-token equivalents per question, including preprocessing for TREEFUSE. Percentages are computed from the displayed rounded proxy totals.

Answer model	Dataset	κ^*	Raw	TREEFUSE	Modeled red. (%)
Qwen3.8-Max $\kappa = 31.502$	Qasper	10.24	163,591	123,773	24.3
	HotpotQA	5.40	424,765	191,750	54.9
	2WikiMQA	6.36	238,521	102,885	56.9
	LaRA	1.66	923,552	532,473	42.3
DeepSeek-V4-Pro $\kappa = 17.149$	Qasper	10.64	87,571	75,833	13.4
	HotpotQA	5.75	219,470	123,776	43.6
	2WikiMQA	6.83	122,023	70,221	42.5
	LaRA	1.82	457,116	273,947	40.1
Kimi-K2.6 $\kappa = 11.089$	Qasper	10.51	56,804	55,741	1.9
	HotpotQA	5.71	142,578	97,075	31.9
	2WikiMQA	6.74	79,792	57,694	27.7
	LaRA	1.76	305,651	190,778	37.6
GLM-5.3 $\kappa = 14.034$	Qasper	10.59	71,650	65,416	8.7
	HotpotQA	5.73	180,233	110,233	38.8
	2WikiMQA	6.75	101,034	64,033	36.6
	LaRA	1.77	385,194	234,980	39.0

For $T_{\text{raw}} > T_{\text{sel}}$, the break-even weight is

$$\boxed{\kappa^* = \frac{C_{\text{pre}}}{T_{\text{raw}} - T_{\text{sel}}}}.$$

This threshold is the primary input-work result: it states how expensive one reader input token must be, relative to one Qwen3.6 preprocessing input token, for the reader context reduction to amortize construction and selection. A deployment can compare the threshold directly with its own reader-to-preprocessor ratio.

For a transparent secondary instantiation, we set κ to activated-parameter ratios obtained from public documentation and model configurations: 11.089 for Kimi-K2.6, 17.149 for DeepSeek-V4-Pro, 31.502 for Qwen3.8-Max, and 14.034 for GLM-5.3. The corresponding activated-parameter counts are 31.686B, 49B, 90.014B, and 40.101B, respectively, against the 2.857B Qwen3.6 reference.

Sensitivity to the reader-weight proxy. For a candidate weight κ, the relative modeled reduction in proxy-weighted successful-path input work is

$$s(\kappa) = \frac{\kappa T_{\text{raw}} - (C_{\text{pre}} + \kappa T_{\text{sel}})}{\kappa T_{\text{raw}}}.$$

Thus $s(\kappa^*) = 0$, and the sign changes exactly at the break-even threshold. Table 14 reports κ^* directly, allowing readers to substitute a different measured or assumed reader-to-preprocessor weight without treating the activated-parameter proxy as exact. Because answer-model tokenizers produce slightly different token counts, each task has a narrow range of model-specific thresholds. The closest case is Qasper with Kimi-K2.6: $\kappa = 11.089$ is only 5.2% above $\kappa^* = 10.51$, so its 1.9% modeled margin is sensitive to modest proxy misspecification. The other task-level threshold ranges are 5.40–5.75 for HotpotQA, 6.36–6.83 for 2WikiMQA, and 1.66–1.82 for LaRA.

Quantitative implications. The break-even thresholds are 5.40–6.83 on HotpotQA and 2WikiMQA, 10.24–10.64 on Qasper, and 1.66–1.82 on LaRA. The low LaRA thresholds reflect reuse of 64 document trees across 887 questions. Under the stated activated-parameter proxies, all 16 combinations have positive modeled reductions in proxy-weighted successful-path input work, ranging from 1.9% to 56.9% (Table 14). For example, the modeled reductions are 54.9% and 56.9% for Qwen3.8-Max on HotpotQA and 2WikiMQA, respectively.

The smallest modeled margin is Qasper with Kimi-K2.6: $\kappa = 11.089$ is only 5.2% above $\kappa^* = 10.51$, producing a 1.9% modeled reduction. Its sign is therefore sensitive to modest misspecification of the activated-parameter proxy. In contrast, LaRA's proxy weights are far above its 1.66–1.82 break-even range. For Qwen3.8-Max on LaRA, the stated proxy corresponds to 391,079 reference input-token equivalents per question after amortized preprocessing.

Accounting scope. Construction and selection inputs use the Qwen3.6 tokenizer. Answer inputs for all four answer models are re-tokenized with their corresponding tokenizers. Answer-input averages use the common questions across three Raw and three TREEFUSE runs. Preprocessing is amortized over the indexed collection in Table 13.

D SOURCE-EVIDENCE ANALYSIS

D.1 EVIDENCE ANNOTATIONS AND SOURCE ALIGNMENT

Qasper uses mapped textual evidence with alternative annotation sets. HotpotQA and 2WikiMQA use mapped supporting passages. The component results are reported in Table 3; the metrics quantify annotated source preservation rather than semantic answer sufficiency.

D.2 EVIDENCE METRIC DEFINITIONS

Complete evidence coverage. Let $\mathcal{G}_i$ contain the eligible evidence sets for question i and $r(u, C_i)$ denote normalized literal coverage of evidence unit u in context C_i. The matcher checks normalized source text and sentence or line segments. We define

$$A_i = \mathbb{1}[\exists G \in \mathcal{G}_i : r(u, C_i) \geq 0.999 \text{ for every } u \in G], \qquad \text{All-hit} = \frac{100}{N} \sum_{i=1}^{N} A_i. \tag{6}$$

Thus, one completely covered alternative is sufficient. Passage-level All-hit measures literal preservation, not the validity of a reasoning chain or semantic answer sufficiency.

Evidence concentration. For purity, selected content is aligned to normalized lexical tokens in the original document. Let P_i be the set of distinct recovered source-token positions, G_i^* the mapped reference-token set, and U_i the number of unmapped output tokens after method-specific content extraction. Then

$$\text{Evidence Purity} = \frac{100}{N} \sum_{i=1}^{N} \frac{|P_i \cap G_i^*|}{\max(1, |P_i| + U_i)}. \tag{7}$$

For Qasper, eligible textual alternatives must map to source tokens, and the scorer chooses the alternative with greatest token recall, breaking ties by source-token intersection-over-union. This rule differs from the literal complete-coverage test used for All-hit. For the multi-hop tasks, reference positions are obtained from the mapped supporting-passage spans. Whole units are matched first, with line-level fallback. Token-pruned outputs use sequence alignment. Structural wrappers are excluded where the parser extracts source content. Recovered source positions are deduplicated, so the denominator is not a literal count of every token sent to the answer model. Unmapped text can include valid paraphrases, and unannotated context can still support answering. Purity is consequently an annotation-based concentration proxy rather than the fraction of all useful information in the output.

D.3 EQUAL-WEIGHT ECP RATIONALE

To summarize completeness and concentration without introducing a tunable trade-off, we use their equal-weight arithmetic mean:

$$\text{ECP} = \frac{1}{2} (\text{All-hit} + \text{Evidence Purity}). \tag{8}$$

All-hit measures whether a complete annotated evidence set is retained, whereas Evidence Purity measures how concentrated the selected content is around mapped evidence. Equal weighting treats these as complementary diagnostic dimensions and avoids choosing a dataset-dependent preference between completeness and concentration. Because both components are percentages, ECP remains on the same 0–100 scale. The same aggregation is applied to every method and dataset; it is a source-evidence diagnostic rather than a model of answer correctness.

E Ablation Details and Case Studies

E.1 Selection, Representation, and Construction Variants

Tables 4–6 use Qwen3.8-Max and a common question set within each comparison over three answer runs. Table 15 summarizes the changes to the full method. BM denotes approximate character-budget matching, not exact per-query token equality.

Table 15: Changes made by each ablation variant.

Variant	Change relative to the full method
Semantic-only	Retain semantic selections and their ancestor context.
Lexical-only (BM)	Use lexical node selection with ancestor closure and an approximately matched character budget.
Semantic-only (BM)	Add fixed-seed random nodes to the semantic selection to approach the full method's context length.
Fixed 200	Replace semantic units with 200-token chunks and apply the same selection framework.
Chonkie	Replace units with the configured Chonkie semantic chunks.
Flat	Retain the semantic units and source text, but remove their hierarchy.
Multi-stage	Replace the constructor with the same-model multi-stage pipeline.
HiChunk adapter	Map HiChunk-derived units into the node format used by our selector rather than using the complete HiChunk retrieval system.

E.2 Tree-Construction Configurations

Table 6 compares representations supplied to the same downstream selection framework. The single-pass and multi-stage constructors both use Qwen3.6-35B-A3B. Single-pass construction generates the hierarchy and node descriptions jointly from the document, as described in Appendix A.3. It denotes a joint generation design, not a guarantee of exactly one request when output validation requires a retry.

Multi-stage construction. The alternative pipeline separates structure extraction, source anchoring, and content enrichment. It first partitions the source into indexed chunks with an 800-token target while respecting passage boundaries, then forms groups of up to 20,000 tokens with one-chunk overlap. An initial call proposes an outline containing titles, hierarchy paths, and chunk indices. Subsequent groups extend the existing outline, with an additional repair step for uncovered chunks when needed. A combined validation and localization stage checks candidate headings against their source chunks and identifies section boundaries. These anchors are mapped to document character positions to assemble the tree and attach local source text. Oversized leaf nodes can be subdivided before bottom-up content enrichment. Leaf descriptions use their own text, whereas internal-node descriptions combine local introductory text with child summaries. The pipeline receives the complete document, but individual calls operate on groups, chunks, or node material. The initial outline call can see the full document when it fits within one group.

Configuration and downstream selection. Validation and localization batch up to five chunks per call. Leaf subdivision uses a 16,000-token threshold and a recursion-depth limit of two. Content enrichment produces summaries and facts, batching up to six nodes with 16,000 characters of material per batch and an 8,000-character per-node limit. Both representations feed the semantic scorer, adaptive cutoff, ancestor completion, and lexical-union procedure. The selector uses the P90 lexical cutoff on both HotpotQA and 2WikiMQA. Node boundaries, metadata, and the selected content can therefore differ despite the shared selection framework.

HiChunk representation adapter. This configuration adapts cached HiChunk-derived segments and hierarchy levels to the common representation interface required by our downstream selector. The adaptation is introduced only for compatibility with the shared selection framework used in this comparison. It therefore evaluates a HiChunk-derived representation with our downstream selection procedure, rather than the complete HiChunk retrieval and Auto-Merge system.

Table 16: Conditional evidence and answer differences with Qwen3.8-Max. Each cell reports ΔSpan Recall in percentage points / ΔQA-F1 in score points. BM denotes approximate character-budget matching.

Full minus	Qasper	HotpotQA	2WikiMQA	Pooled
Semantic-only	+1.14 / +0.11	+1.86 / +0.40	+0.53 / +0.56	+1.17 / +0.36
Semantic-only (BM)	+1.14 / −1.14	−0.07 / +2.42	+0.22 / +2.76	+0.41 / +1.42
Lexical-only (BM)	+16.45 / +0.60	+18.96 / +4.17	+28.46 / +8.14	+21.50 / +4.45

E.3 Conditional Evidence and Answer Analysis

Table 16 pairs each variant's saved context with its Qwen3.8-Max answers. Span Recall measures length-weighted literal coverage of annotated source units. Among alternative evidence sets, the scorer prefers complete coverage, then the highest weighted coverage. QA-F1 is averaged over three runs for each question. Each reported difference is Full minus the named variant.

Semantic-only (BM) augments the original semantic selections with fixed-seed random nodes to approximate the full method's context length. Lexical-only (BM) uses lexical node selection with ancestor closure. The generation-setting differences recorded in Appendix B apply to these comparisons as well. They describe the available configurations rather than isolating every change from answer-generation effects.

E.4 Semantic Support and Lexical Supplementation Cases

The examples below are drawn from saved Qwen3.8-Max outputs and their paired contexts. They illustrate distinct selection behaviors rather than estimating population-level success rates. Span Recall and QA-F1 are percentages and three-run means, respectively. Displayed source excerpts are verbatim.

Lexical supplementation: recovering data provenance. A Qasper question asks, "How is the dataset of hashtags sourced?" The source identifies "1,268 randomly selected tweets in the Stanford Sentiment Analysis Dataset" and "all 12,594 unique English hashtags and their associated tweets from the same Stanford dataset." Compared with Semantic-only (BM), Full raises Span Recall from 0 to 100 and mean QA-F1 from 0 to 71.11. All three Semantic-only (BM) answers are "unanswerable", whereas all three Full answers identify the Stanford Sentiment Analysis Dataset. This illustrates the value of retaining source-specific provenance in the selected context.

Semantic support: connecting a mission to its destination. A HotpotQA question asks, "Gary L. Bennett was a part of the space missions that have a primary destination of what celestial body?" The source links Bennett to the Ulysses mission and states that Ulysses "was a robotic space probe whose primary mission was to orbit the Sun and study it at all latitudes." Compared with Lexical-only (BM), Full raises Span Recall from 2.10 to 82.89. All three lexical-only answers name Jupiter, whereas all three Full answers name the Sun, changing mean QA-F1 from 0 to 100. The example illustrates how the selected source context can connect a named participant to the mission target.

Evidence enrichment without an answer-score change. Another HotpotQA question asks, "Which California band, whose debut album Adrenaline appeared in 1995, has been referred to as 'the Radiohead of metal'?" The source identifies Deftones and states, "The band's experimental nature has led some critics to describe them as the Radiohead of metal." lexical-only (BM) and Full both answer Deftones in all three runs. Full increases Span Recall from 87.28 to 96.34 while mean QA-F1 remains 100. Thus, identical correct answers can arise from contexts with different annotated-evidence coverage. These are examples from the recorded configurations described in Appendix B.2, not isolated causal tests.

E.5 Semantic Adequacy versus Boundary Exactness

Routing metadata and source evidence. Generated boundaries, hierarchy, summaries, and facts can affect which source regions are selected. The relevant distinction is between *evidence containment* and exact recovery of annotation boundaries: the tree organizes source regions for retrieval rather than reproducing a unique gold segmentation. Titles and structural paths provide navigation, while summaries and facts guide selection rather than substitute for source evidence in the answer context. The source matcher anchors nodes to document text; when both summary and facts are absent, scoring uses a source-text prefix (Appendix A.3). The following case examines whether a node broader than individual evidence units can still retain their support.

A coherent region containing the annotated mechanism. The Qasper example `qasper-125-5724368043143d51-q151` asks, *"How are weights dynamically adjusted?"* One annotated alternative evidence set contains four units, all mapped to node `n8`. These units explain why easy-negative examples dominate training, the modification that multiplies soft probability p by $(1 - p)$, how this factor changes during training to down-weight easy examples, and its connection to focal loss. The node contains the mechanism together with its motivation and interpretation, rather than assigning a separate annotation-aligned boundary to each unit. The four-unit set is the alternative used for the case metrics below, not the union of all annotations for this question; matching follows Appendices D.1 and D.2.

Retained evidence and downstream answers. Table 17 compares the saved Lexical-only (BM) and Full contexts. Full retains **all four evidence units** in the source region, raising Span Recall from 0 to 100 and mean QA-F1 from 17.57 to 35.33. The Lexical-only context omits all four units. Thus, complete recovery of this annotated support occurs without exact unit-level segmentation. The QA-F1 gain accompanies improved evidence coverage; it does not imply a perfectly correct answer or isolate boundary placement as its cause.

Table 17: Evidence retention for the Qasper weight-adjustment case. Span Recall and QA-F1 use a 0–100 scale; QA-F1 averages three Qwen3.8-Max answer runs. BM matches mean character budgets, not individual context lengths.

Condition	Evidence units	Span Recall	All-hit	Mean QA-F1	Context chars.
Lexical-only (BM)	0/4	0.00	No	17.57	5,050
Full (TreeFuse)	4/4	100.00	Yes	35.33	12,552

Implication for construction reliability. This case establishes a narrower requirement than boundary-perfect reconstruction: **a coherent source node can preserve the required evidence without matching its annotation boundaries**. For source-grounded selection to work, the relevant text must remain represented in selectable nodes, those nodes must be retrieved, and their original text must reach the reader. Boundary or metadata differences need not remove support when these conditions remain satisfied. Conversely, source grounding does not prevent incorrect metadata or structure from causing a selection miss. This is an observed containment case, not a controlled corruption experiment or an estimate of tree-construction error rates. It supports separating routing metadata from answer evidence, rather than treating arbitrary construction errors as harmless.