BEAR: Budgeted Evidence Allocation for Multi-Document Reasoning

Anonymous ACL submission

Abstract

We argue that multi-document reasoning is constrained not only by how much text a model can read, but also by how limited query-time evidence budget is allocated across documents and semantic granularities. Full-context inference exposes the model to broad evidence non-selectively and at high per-query cost, while flat chunk retrieval often returns locally relevant passages that are weakly organized for cross-document synthesis. We present **BEAR**, a framework for structured evidence allocation that builds hierarchical semantic indices offline and performs coarse-to-fine evidence access at query time through complementary *exploration* and *recovery* paths. This coarse-to-fine design can be viewed as structured evidence allocation under a fixed evidence-context budget. Across synthetic and real-world benchmarks, BEAR performs particularly strongly on DragonBall, remains competitive with strong retrieval-based baselines on HotpotQA, and yields the best retrieval-based result on 2Wiki under our evaluated protocol, while operating under substantially smaller *query-time evidence budgets* than the reported long-context references. Additional analyses suggest that the gains are associated with hierarchy as an allocation substrate together with complementary exploration and recovery, rather than semantic chunking alone.

1 Introduction

Multi-document reasoning requires more than simply exposing an LLM to a large amount of text. In realistic settings, systems must answer under limited inference-time resources, including constraints on retrieved evidence, generator context, and query-time compute. A central challenge is therefore how limited *evidence budget* should be allocated across documents and semantic granularities. Even with long context windows, relevant evidence may be dispersed across documents or buried in distracting context, while flat retrieval often returns passages that are locally similar to the query but weakly organized for cross-document synthesis (Liu et al., 2024; Kamradt, 2023; Comanici et al., 2025; Bai et al., 2024; Ram et al., 2023; Shi et al., 2023).

Figure 1 illustrates a financial comparison query that requires both coarse evidence allocation and localized evidence recovery. Flat retrieval tends to over-retrieve weakly useful chunks, while naive long-context inference leaves evidence prioritization and localization implicit inside generation. A structured system can instead expose coarse semantic abstractions for global allocation and refine to local evidence only when needed.

Motivated by this, we propose **BEAR**, a framework for budget-aware evidence allocation in multi-document reasoning. BEAR organizes documents offline into hierarchical semantic trees and accesses them online through two complementary paths (Figure 2): an *exploration path*, implemented via LLM-guided semantic selection over high-level summaries, and a *recovery path*, implemented via TreeExpansion for localized evidence recovery. Hierarchy provides the substrate for coarse-to-fine query-time access: a small coarse-grained budget first identifies promising semantic regions, after which finer-grained evidence access focuses on detailed supporting evidence. This budget compression helps preserve global coverage without paying the full cost of exhaustive flat retrieval or long-context processing. Relative to flat retrieval, this preserves coarse semantic abstractions and multi-granularity access; relative to naive long-context inference, it makes evidence organization explicit.

BEAR formulates evidence access over hierarchical semantic indices under a fixed query-time evidence budget. This yields two operating regimes: **BEAR(docs)** for budget-efficient document-level fusion and **BEAR(nodes)** when coarse evidence exceeds the budget or finer evidence discrimination is needed. Our contributions are threefold: (1) we frame multi-document reasoning as a problem

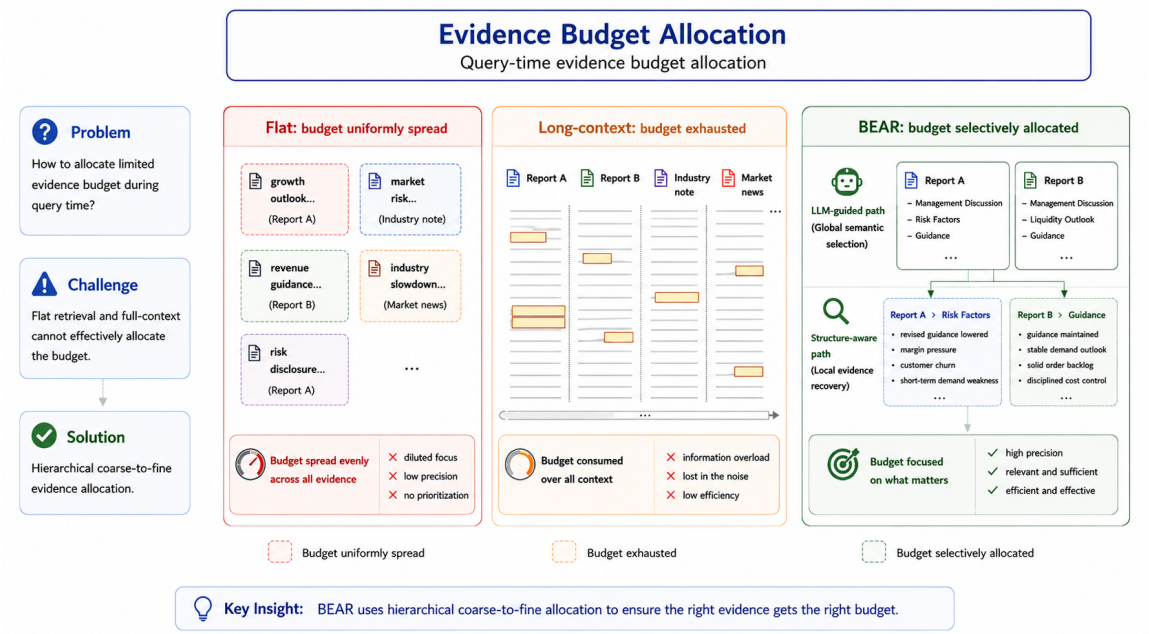

Figure 1: Motivating example for complementary exploration and recovery under a limited evidence budget.

of budgeted evidence allocation across semantic granularities; (2) we introduce a coarse-to-fine evidence allocation framework with complementary exploration and recovery paths, in which hierarchy serves as a mechanism for budget-aware evidence access rather than only an indexing structure; and (3) we provide controlled evaluations across synthetic, real-world, and downstream agentic settings suggesting that this design improves over flat retrieval in our controlled ablations and offers a favorable quality–cost operating point across document-level and node-level access regimes.

2 Related Work

Long-context LLMs can process substantially more text, but longer context does not reliably yield stronger knowledge-intensive reasoning and increases computational cost (Liu et al., 2024; Kamradt, 2023; Comanici et al., 2025; Bai et al., 2024). This has sustained interest in retrieval-based systems, especially when evidence must be selected rather than merely concatenated (Lewis et al., 2020; Gao et al., 2024).

Structured retrieval methods improve on flat chunks by introducing graph or hierarchical organization. Graph-based systems such as GraphRAG, LightRAG, and HippoRAG organize evidence around entities and relations, while hierarchical methods such as RAPTOR and HiRAG build tree-like abstractions over documents (Edge et al., 2024; Guo et al., 2024; Gutiérrez et al., 2024; Sarthi et al., 2024; Huang et al., 2025; Jin et al., 2025). BEAR builds on this line of work by focusing on query-time access under a fixed evidence budget: the emphasis is not only hierarchy construction, but coarse-to-fine traversal and fusion under budget constraints. Related systems such as EfficientRAG and Tree of Reviews similarly highlight the value of efficient or iterative structured retrieval for multi-hop reasoning (Zhuang et al., 2024; Jiapeng et al., 2024).

A third line of work integrates LLM reasoning more directly into retrieval, including iterative retrieval-and-reasoning pipelines and agentic systems that interleave reasoning with tool use (Trivedi et al., 2023; Press et al., 2023; Asai et al., 2023; Jiang et al., 2023; Yao et al., 2023; Nakano et al., 2022; Schick et al., 2023). BEAR sits between static structured retrieval and fully reactive retrieval agents: it organizes document semantics offline into hierarchical forests, then navigates that structure online through budget-aware evidence allocation and selective refinement.

Positioning BEAR. Relative to prior hierarchical or tree-based RAG, BEAR differs in emphasis: it studies how a hierarchical semantic index is used at query time under a fixed evidence budget rather than hierarchy construction in isolation. Coarse document-level evidence is preserved when sufficient, node-level refinement is invoked only when

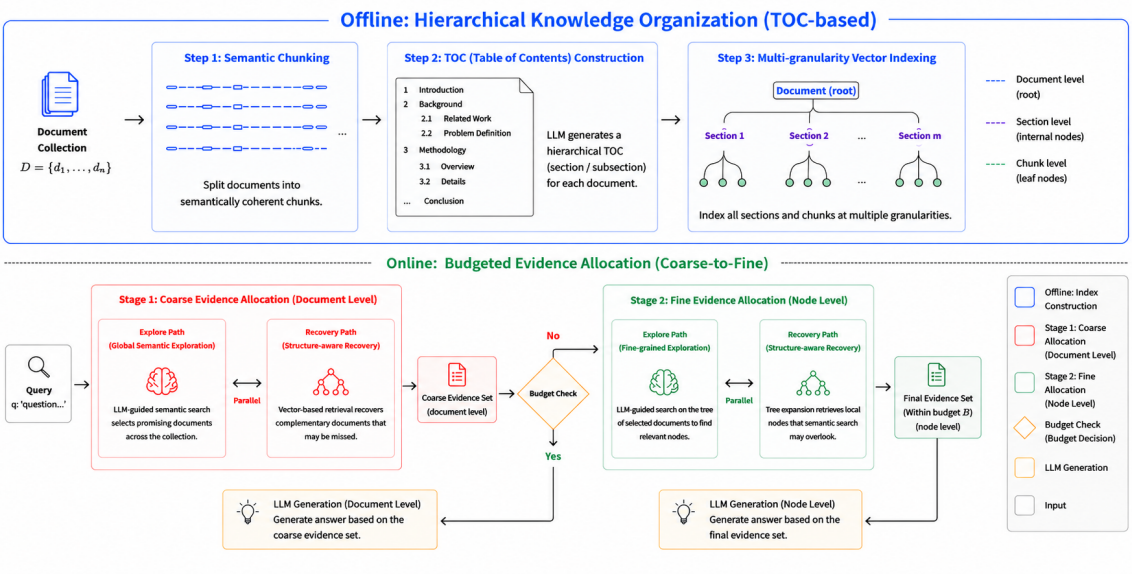

Figure 2: Overview of BEAR. Offline, it builds hierarchical semantic indices and multi-granularity representations. Online, it progressively allocates evidence through complementary exploration and recovery under a fixed evidence budget, refining to finer-grained evidence only when needed.

finer-grained aggregation is required, and the node-level stage combines an exploration path with a recovery path rather than a single traversal mechanism. Relative to flat retrieval and monolithic long-context inference, this design allocates a limited query-time evidence budget more selectively while making evidence organization explicit.

3 Method

3.1 Overview

BEAR uses a semantic forest (one tree per document) $\mathcal{F} = \{T_1, \ldots, T_N\}$ as the substrate for budget-aware query-time access over a document collection $\mathcal{D} = \{d_1, \ldots, d_N\}$. **Offline**, an LLM segments each document into coherent chunks, organizes them hierarchically, and produces summary-bearing internal nodes for high-level retrieval. Because evidence granularity itself becomes a resource-allocation decision under a fixed query-time budget, hierarchical representations provide a natural substrate for allocating limited evidence across abstraction levels. **Online**, BEAR first allocates coarse evidence candidates through a complementary exploration–recovery stage, then optionally refines them at node level if the coarse result exceeds evidence-context budget.

BEAR is designed to support both *global semantic exploration* and *localized evidence recovery*: one path uses LLM reasoning over hierarchical summaries to identify relevant regions, while the other uses dense similarity and structural propagation to recover supporting evidence that a purely reasoning-driven path may miss.

3.2 Offline Hierarchical Knowledge Organization

For each document d_i, BEAR first performs LLM-guided semantic chunking instead of fixed-length splitting, preserving discourse coherence and aligning chunk boundaries with semantic units. Let

$$\mathcal{C}_i = \mathrm{LLM}_{\mathrm{segment}}(d_i) = \{c_1, c_2, \ldots, c_{m_i}\} \quad (1)$$

denote the resulting chunk sequence.

BEAR then constructs a semantic tree

$$T_i = \mathrm{LLM}_{\mathrm{structure}}(\mathcal{C}_i \mid d_i), \quad (2)$$

whose internal nodes provide topic-level abstractions (e.g., titles and summaries) for high-level retrieval, while the leaves retain the underlying chunk content needed for answer generation.

To support multi-granularity retrieval, BEAR builds separate embeddings for internal and leaf nodes. For a non-leaf node v, the embedding is computed from the path-level table-of-contents signal together with its summary:

$$\mathbf{e}_v = \mathrm{Embed}(\mathrm{toc_path}(v) \oplus \mathrm{summary}(v)). \quad (3)$$

For a leaf node c, we embed its original chunk content:

$$\mathbf{e}_c = \mathrm{Embed}(\mathrm{content}(c)). \quad (4)$$

These vectors form the basis of the dense retrieval path at both document and node levels.

3.3 Budget-Aware Evidence Allocation

Coarse evidence access. At query time, BEAR first performs coarse evidence access through two complementary paths. The first, the *exploration path*, exposes only high-level non-leaf summaries to the LLM, allowing it to reason over compact semantic abstractions. The second, the *recovery path*, performs dense retrieval over the indexed nodes. Their union forms a deduplicated candidate set $\mathcal{D}_{\text{fusion}}$, combining semantic reasoning with embedding-space coverage.

Budget-aware coarse-to-fine routing. Here, the query-time budget reflects the limited evidence bandwidth available to expose supporting evidence before generation. In practice, we instantiate this constraint using a token budget over the final retrieved evidence context, denoted by $B_{\max}$. BEAR first checks whether the fused document-level evidence fits within this budget. If so, BEAR directly returns the selected evidence context; otherwise, BEAR proceeds to a finer node-level stage. In the experiments, **BEAR(docs)** returns the fused document-level context, while **BEAR(nodes)** performs additional node-level exploration–recovery refinement when coarse evidence exceeds the budget or finer evidence discrimination is needed.

Fine-grained evidence refinement. When refinement is needed, BEAR again uses two paths. The first, the exploration path, asks the LLM to navigate hierarchical summaries within the selected documents and identify promising semantic nodes. The second, the recovery path, applies TreeExpansion, a lightweight tree-restricted evidence recovery mechanism that supplements direct query-node similarity with signals inherited from ancestors and aggregated from children. For a node v, we score it as:

$$S(v) = \alpha S_{\text{sim}}(v) + \beta S_{\text{inh}}(v) + \gamma S_{\text{child}}(v), \quad (5)$$

where $S_{\text{sim}}(v)$ is direct query–node similarity, $S_{\text{inh}}(v)$ propagates relevance from ancestors, and $S_{\text{child}}(v)$ aggregates support from descendants. Unless otherwise specified, we use $\alpha = \beta = \gamma = \frac{1}{3}$ as a simple untuned default that does not require query-type prediction or additional online adaptation. Appendix Table 10 shows that some query categories prefer different weightings, but we keep the equal-weight setting to preserve a fixed and lightweight retrieval policy. This scoring rule complements summary-level semantic guidance under the same evidence-budget constraint. The final evidence list is deduplicated and truncated to satisfy the budget.

4 Experimental Setup

Datasets. We evaluate BEAR on synthetic and real-world multi-document reasoning benchmarks. For synthetic evaluation, we use DragonBall (Zhu et al., 2025), including settings stressing evidence integration, hallucination, and multilingual retrieval. DragonBall's use of LLM-synthesized knowledge provides a cleaner signal for retrieval quality: since modern LLMs are trained on Wikipedia, real-world benchmarks such as HotpotQA and 2Wiki risk conflating retrieval gains with parametric knowledge recall, whereas synthetic knowledge isolates retrieval as the primary variable. For real-world multi-hop reasoning, we use HotpotQA (Yang et al., 2018) and 2Wiki (Ho et al., 2020), adapted for retrieval over candidate document collections. We additionally use BrowseComp-plus (Chen et al., 2025) as a downstream agentic benchmark to test whether retrieval gains transfer beyond benchmark-style QA.

Baselines. We compare against sparse and dense retrieval baselines (BM25 and BGE-M3 (Chen et al., 2024)), structure-enhanced baselines (RAPTOR, LongRefiner (Jin et al., 2025), and HippoRAG2 (Gutiérrez et al., 2024; Gutiérrez et al., 2025)), a same-generator full-context control (Qwen3-32B (Team, 2025)), and long-context references (Gemini-2.5-Flash and Gemini-2.5-Pro). RAPTOR is included as a well-known tree-structured baseline closely related to BEAR's hierarchical indexing setting, while HippoRAG2 serves as a strong graph-based structured retriever.

For reproducibility, we use public implementations whenever available: BM25 and BGE-M3 are based on FlashRAG (Jin et al., 2024), while RAPTOR, LongRefiner and HippoRAG2 use their official repositories. Unless a baseline has an intrinsic model dependency, offline and retrieval-time LLM components are aligned to DeepSeek-V3.2 (DeepSeek-AI et al., 2025) and the final generator is fixed to Qwen3-32B; LongRefiner is the main exception because its preprocessing relies on a separately trained helper model. Dense retrieval is aligned to BGE-M3, with BGE-Reranker-

Table 1: Results on DragonBall, HotpotQA, and 2Wiki (mean ± standard deviation). **BEAR(docs)** returns fused document-level context; **BEAR(nodes)** adds node-level refinement. Retrieval-based methods share the same Qwen3-32B generator and, where applicable, DeepSeek-V3.2 for offline and retrieval-time steps; LongRefiner is the main exception. For DragonBall, we report Recall, Completeness, Hallucination, and Irrelevance; Recall is omitted for Gemini because these references receive full source documents rather than retrieved evidence.

Method	Synthetic Knowledge				Real-World Knowledge			
	DragonBall				HotpotQA		2Wiki	
	Recall(%) ↑	Comp.(%) ↑	Hall.(%) ↓	Irr.(%) ↓	EM(%) ↑	F1(%) ↑	EM(%) ↑	F1(%) ↑
Matched Retrieval Baselines								
BM25	29.17 ± 1.06	35.13 ± 1.84	32.55 ± 2.40	32.25 ± 1.95	20.67 ± 0.76	32.69 ± 0.53	10.33 ± 1.44	18.12 ± 1.32
BGE-M3	58.17 ± 1.91	59.56 ± 2.45	28.81 ± 2.26	11.54 ± 1.55	22.83 ± 2.08	36.25 ± 2.83	15.83 ± 1.26	26.94 ± 1.17
RAPTOR	31.93 ± 1.66	59.78 ± 2.70	26.50 ± 2.48	13.67 ± 1.29	29.33 ± 1.26	44.72 ± 1.70	23.50 ± 1.00	36.96 ± 1.30
LongRefiner	29.24 ± 2.32	43.86 ± 1.98	34.88 ± 2.25	21.26 ± 1.87	22.33 ± 0.85	33.97 ± 0.71	9.00 ± 0.87	13.16 ± 0.83
HippoRAG2	68.80 ± 1.75	73.37 ± 2.24	19.55 ± 1.84	6.93 ± 0.93	**33.83 ± 0.58**	**48.99 ± 1.22**	29.67 ± 0.76	44.19 ± 0.73
Full-context Controls / References								
Gemini-2.5-Flash	–	88.62 ± 0.34	6.30 ± 0.14	5.09 ± 0.04	29.83 ± 1.31	45.96 ± 1.06	38.67 ± 0.24	55.21 ± 0.27
Gemini-2.5-Pro	–	90.86 ± 0.07	5.46 ± 0.24	3.67 ± 0.09	41.74 ± 0.33	56.66 ± 0.90	53.01 ± 1.41	64.69 ± 0.72
Our Proposed Method								
BEAR(docs)	**85.55 ± 0.47**	**91.86 ± 1.32**	**5.64 ± 1.02**	**2.40 ± 0.49**	31.00 ± 1.15	45.80 ± 1.18	**34.63 ± 2.17**	**48.02 ± 1.45**
BEAR(nodes)	84.25 ± 0.84	88.64 ± 1.52	7.60 ± 1.12	3.71 ± 0.86	33.50 ± 1.15	48.44 ± 0.54	33.00 ± 1.73	46.30 ± 1.48

v2-M3 (Li et al., 2023) used when reranking is required. Other shared settings appear in Appendix Table 7.

Comparison protocol. We report four analyses: matched retrieval baselines, a same-generator full-context control, controlled ablations, and long-context references. In the matched retrieval comparisons, we keep the answer generator fixed and, where applicable, also align the offline and retrieval-time LLMs so differences are primarily attributable to retrieval policy rather than generation or auxiliary models. Qwen3-32B serves as the same-generator full-context control, while Figure 3 and the progressive ablations isolate the effects of semantic chunking, hierarchical indexing, and bi-path retrieval. Gemini-2.5-Flash and Gemini-2.5-Pro are contextual long-context references rather than matched retrieval baselines. Across retrieval-based methods, the final evidence context passed to the common Qwen3-32B generator is aligned to approximately 4K tokens whenever applicable, reducing confounding from unequal generator context length. When public baseline defaults yield substantially shorter generator-side contexts, we preserve the original retrieval logic and adjust only the final evidence packing or truncation to better match the shared budget. This keeps the comparison focused on retrieval behavior while controlling for unequal generator-side context length; accordingly, our main claim is about the operating point achieved under a shared-generator, budget-matched retrieval protocol rather than universal superiority across model families or serving stacks. For BrowseComp-plus, we keep the agent policy model fixed and replace only the retriever with BEAR.

Metrics and protocol. For DragonBall, we report Recall, Completeness, Hallucination, and Irrelevance, following (Zhu et al., 2025). For HotpotQA and 2Wiki, we report EM and F1 using the official HippoRAG2 scripts. Unless otherwise noted, we report mean and standard deviation over 3 repeated evaluation runs under a fixed evaluation setup. We also analyze token cost, end-to-end latency, and the contribution of bi-path retrieval; robustness of TreeExpansion appears in Appendix Table 10.

5 Results

We organize the results around three questions: whether BEAR improves the budget-constrained operating point, where the gains come from, and whether they persist in matched, cost, and transfer analyses.

5.1 Main Results

Table 1 shows that BEAR performs particularly strongly on DragonBall among the evaluated retrieval-based methods: both variants substantially outperform the matched retrieval baselines, and **BEAR(docs)** provides the strongest retrieval-based trade-off in the table. On HotpotQA, **BEAR(nodes)** remains close to HippoRAG2, while on 2Wiki **BEAR(docs)** gives the strongest retrieval-based result under the evaluated protocol. We include Qwen3-32B as a full-context control and

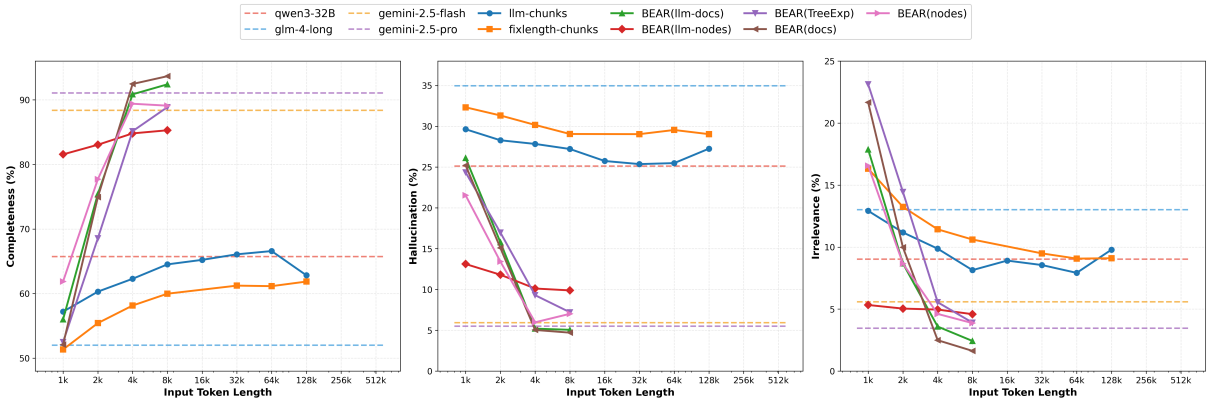

Figure 3: Performance under different evidence-context budgets. Chunk-based controls use fixed-size chunking (*fixlength-chunks*) or semantic chunking (*llm-chunks*) within the same Standard RAG pipeline; the remaining curves are BEAR variants. In the evaluated setting, structured evidence allocation yields a stronger completeness–faithfulness trade-off than the chunk-based controls at much smaller budgets.

Gemini-2.5-Flash/Pro as long-context references rather than matched retrieval baselines.

On the synthetic benchmark, the clearest advantage is the coverage–faithfulness trade-off. On DragonBall, **BEAR(docs)** reaches 85.55% Recall and 91.86% Completeness while keeping Hallucination and Irrelevance at 5.64% and 2.40%, respectively. Table 2 further shows that, under the same Qwen3-32B generator, passing the full document set directly to the model sharply reduces answer quality, suggesting that the gains are associated with evidence organization and selection rather than with the answer model alone. On the real-world multi-hop benchmarks, the two variants reflect different operating regimes. **BEAR(nodes)** is competitive on HotpotQA and remains close to HippoRAG2; HippoRAG2's KG construction explicitly links entities across documents, giving it a structural advantage on HotpotQA's entity-centric multi-hop design, while BEAR's per-document semantic trees do not model cross-document entity links. On 2Wiki, **BEAR(docs)** performs best among the compared retrieval-based methods. **BEAR(docs)** is preferable when coarse-grained evidence allocation already yields a coherent evidence set, whereas **BEAR(nodes)** is preferable when finer-grained evidence aggregation improves precision.

Figure 3 further clarifies these gains under different evidence-context budgets. Semantic chunking consistently improves over fixed-size chunking, but both chunk-based variants remain well below the hierarchical BEAR variants, especially in the low- and medium-budget regimes. Under very small budgets (1K–2K), **BEAR(llm-nodes)** is the strongest node-level variant. At moderate budgets (4K–8K), the structured variants become competitive with the reported long-context references in this contextual comparison while using 4K–8K retrieved tokens rather than 512K-token full-context inputs. Together, these controls suggest that the gains are associated with hierarchy as an allocation substrate together with complementary exploration–recovery evidence allocation rather than with semantic chunking alone or simply exposing more text.

Table 2: DragonBall full-context control under the same Qwen3-32B generator. Entries report mean ± standard deviation over repeated runs. The comparison isolates the effect of passing the full document set directly to the generator versus structured evidence selection.

Method	Comp. (%)	Hall. (%)	Irr. (%)
Qwen3-32B	66.86 ± 0.30	25.36 ± 0.33	7.70 ± 0.01
BEAR(docs)	**91.86 ± 1.32**	**5.64 ± 1.02**	**2.40 ± 0.49**
BEAR(nodes)	88.64 ± 1.52	7.60 ± 1.12	3.71 ± 0.86

5.2 Why exploration–recovery helps

We focus on the node-level stage, where the interaction between the exploration and recovery paths is most directly observable. Table 3 shows that neither path is uniformly strongest across answerable query types: the recovery path (TreeExpansion) is stronger on factual and temporal questions (FQ, TSQ), where localized or order-sensitive evidence matters, while the exploration path (LLM-guided selection) is stronger or competitive on synthesis-heavy categories such as information integration (IIQ), numerical comparison (NCQ), multi-hop rea-

soning (MRQ), and summarization (SQ). Bi-path fusion improves over the stronger single path on every answerable query type, with gains from +2.08 to +8.99 points and an average of +6.01. Table 12 shows the paths are not redundant: 80.7% of supported evidence is recovered by both, while 10.3% is found only by TreeExpansion and 6.1% only by the LLM-guided path. Multi-hop Reasoning is the clearest hard case, with overlap dropping to 72.3% and the uncovered portion rising to 8.4%. Error analysis shows that the "neither" cases are concentrated in the medical subdomain and mostly involve deep leaf nodes for multi-hop or comparison queries. On the exploration path, the main failure is conclusion-node selection bias: the model often selects the semantically closest summary node whose subtree does not cover the evidence-bearing sibling branches required for the answer. On the recovery path, misses arise from entity-sparse retrieval in a homogeneous corpus or from cross-branch truncation under the fixed evidence budget; Appendix A.1 provides the full breakdown.

The progressive ablation supports a staged attribution: semantic chunking helps, evidence access over the hierarchical index provides the largest improvement over flat retrieval in the evaluated ablation, and complementary bi-path evidence access supplies the final increase in evidence coverage. The largest jump comes from moving from leaf-only evidence access to tree-aware evidence access (50.55%→87.91%), indicating that the hierarchical index provides a substantially stronger allocation substrate than flat retrieval built on the same semantically chunked leaves. This gain reflects the value of the LLM-constructed hierarchical semantic index, including internal summary-bearing nodes and hierarchical organization, rather than a control that isolates bare tree topology. On top of this substrate, TreeExpansion, LLM-guided evidence selection, and especially their fusion provide further gains by recovering complementary evidence under the same budget. Appendix Table 13 further shows that swapping the offline and online LLMs mainly changes the relative strength of the single-path components, especially the online exploration path, while the final bi-path system remains comparatively stable. Full query-type results appear in Appendix Table 11, and Appendix Table 9 shows that this complementarity remains stable across domains and languages. The magnitude of the leaf-to-tree jump also suggests a structural prerequisite: gains from hierarchical indexing are largest when document organization is rich enough for LLM-built parent summaries to be informative, and may shrink on corpora where that structure is weak or absent.

Table 3: Fine-grained evidence analysis across query types on answerable questions. Gain reports the improvement of bi-path fusion over the stronger single path; higher is better.

Type	LLM-guided (%)	TreeExp. (%)	Bi-path (%)	Gain
FQ	87.50	95.42	**99.17**	+3.75 pp
IIQ	88.39	90.08	**96.68**	+6.60 pp
NCQ	90.35	90.07	**97.22**	+6.87 pp
TSQ	89.65	95.42	**97.50**	+2.08 pp
MRQ	87.12	86.24	**96.11**	+8.99 pp
SQ	91.35	91.55	**99.31**	+7.76 pp

Table 4: Progressive ablation of node-level evidence coverage. All variants use the same LLM-based semantic chunking; Leaf-only restricts evidence access to leaf chunks, while Tree-aware uses hierarchy-aware evidence access over the hierarchical index. Full query-type results appear in Appendix Table 11.

Variant	Hier.	LLM-guided	TreeExp.	Recall (%)
Leaf-only	×	×	×	50.55
Tree-aware	✓	×	×	87.91
Tree + LLM-guided	✓	✓	×	89.25
Tree + TreeExp.	✓	×	✓	91.55
BEAR(nodes)	✓	✓	✓	**98.19**

5.3 Transfer to agentic retrieval

We additionally evaluate BEAR on BrowseComp-plus, an agentic search-and-synthesis benchmark over large document collections. With the agent policy model fixed, replacing only the retriever with BEAR improves accuracy from 44.46% to 66.60% and recall from 62.32% to 76.60%, suggesting that the retrieval gains can transfer beyond benchmark-style QA in this controlled agent replacement setting. Appendix Table 8 also shows that BEAR reduces the number of search calls relative to the original retriever only in the reported Tongyi-DeepResearch comparison setup.

5.4 Cost and Latency Analysis

We separate one-time indexing workload from online serving cost because systems distribute preprocessing and serving effort differently.

Online serving cost. Table 5 reports online token usage and latency under the same budget-matched protocol as the main retrieval comparisons, so quality and efficiency are evaluated under the same

Table 5: Online serving token usage and latency. Retrieval-based systems share the same Qwen3-32B generator; Gemini is a long-context reference. Token counts are in thousands (K) and averaged per query.

	Retrieval (K tokens)		Generation (K tokens)		Latency (s)		
Method	In	Out	In	Out	Retrieval	Generation	Total
BEAR(nodes)	27.95	0.02	**3.52**	**0.14**	6.81	**16.71**	**23.52**
Standard RAG	**0**	**0**	4.21	0.89	**0.11**	27.37	27.48
HippoRAG2	2.91	0.04	4.21	0.79	7.15	25.42	32.57
LongRefiner	129.25	0.59	3.94	0.63	17.16	20.90	38.06
RAPTOR	**0**	**0**	3.79	1.01	5.47	21.40	26.87
Gemini-2.5-Flash	0	0	397.75	0.36	0	19.23	19.23
Gemini-2.5-Pro	0	0	397.75	0.80	0	19.34	19.34

setting. Within the retrieval-based group, all final answers are generated by the same Qwen3-32B backbone. Under this controlled protocol, BEAR provides a favorable online quality–latency profile among the retrieval-based methods we evaluate. Relative to LongRefiner and HippoRAG2, it substantially reduces total latency, and relative to Standard RAG and RAPTOR, it shifts more computation into retrieval while still achieving lower end-to-end latency. Overall, these results suggest that BEAR pays a moderate retrieval-stage cost for a stronger end-to-end quality–latency trade-off than the alternative retrieval-based baselines we compare.

We also report Gemini-2.5-Flash and Gemini-2.5-Pro as long-context references. Their online burden falls almost entirely on generation, reaching 397.75K input tokens on average, and their latency values are contextual because they rely on different long-context models and serving stacks. These measurements complement, rather than redefine, BEAR's main query-time budget: the token budget on retrieved evidence before final generation.

Offline indexing workload. Offline preprocessing cost is reported as token volume because the indexing pipelines rely on different auxiliary models. BEAR, RAPTOR and HippoRAG2 use DeepSeek-V3.2 for LLM-assisted index construction, whereas LongRefiner relies on a separately trained Qwen2.5-3B-Instruct helper model. Under this view, BEAR requires substantially fewer offline input tokens than HippoRAG2 (0.95M vs. 3.74M), but more than RAPTOR (0.46M) and LongRefiner (0.39M). At the same time, BEAR produces substantially more output tokens because it builds a summary-rich hierarchical semantic index rather than a lighter preprocessing artifact. Most of BEAR's offline workload comes from semantic chunking and tree construction. Maintaining the hierarchy under document updates introduces additional systems overhead not captured by token volume alone.

Table 6: Offline indexing token workload. Token counts are in millions (M). LongRefiner uses a separate Qwen2.5-3B-Instruct helper model in preprocessing.

Method / stage	Input (M)	Output (M)
BEAR(nodes)	0.95	2.65
chunking	0.43	1.48
tree building	0.53	1.17
HippoRAG2	3.74	0.88
LongRefiner	0.39	0.22
RAPTOR	0.46	0.09

6 Conclusion

We presented **BEAR**, a framework for budgeted evidence allocation in multi-document reasoning that combines hierarchical semantic indexing, complementary exploration–recovery evidence allocation, and budget-aware query-time access under a fixed evidence-context budget. Across the evaluated benchmarks, BEAR performs particularly strongly on DragonBall, remains competitive on HotpotQA, and yields the best retrieval-based result on 2Wiki under our matched protocol, while using smaller *query-time evidence budgets* than the reported long-context references. Coarse-grained evidence access provides a favorable completeness–faithfulness trade-off, while node-level refinement is most useful when finer-grained evidence aggregation is needed. Overall, our findings suggest that increasing available context alone is insufficient; evidence exposure itself must be selectively organized. BEAR occupies a favorable operating regime for semantically organized corpora under constrained query-time budgets, with tradeoffs in offline indexing cost, online latency, and dependence on semantically structured corpora.

Limitations

BEAR is most effective when the document collection has enough semantic structure for hierarchical organization to be informative. On highly noisy, weakly structured, or rapidly changing corpora, the hierarchy may become less reliable as an allocation substrate, and the gains from selective evidence allocation may diminish. The method also incurs nontrivial offline indexing cost and additional system complexity relative to standard flat RAG pipelines, since it requires semantic chunking, hierarchical tree construction, and multi-granularity indexing before query-time retrieval. In dynamic settings, practical deployment may require subtree reconstruction, index maintenance under document drift, and preserving summary consistency across abstraction levels. In our current implementation, updates are localized at the document level: when a document changes, we rerun semantic chunking, tree construction, node embedding, and the corresponding vector index updates only for that document, while leaving unchanged documents untouched. Finally, BEAR improves evidence selection and organization, but it does not by itself eliminate downstream reasoning errors of the generator. In practice, BEAR is most attractive when documents have reusable semantic organization, evidence must be selected under a fixed query-time evidence budget, and the application can tolerate offline preprocessing and structure maintenance. These tradeoffs make BEAR particularly suitable when improved budgeted reasoning quality justifies preprocessing and structured indexing, and less suitable for fully general-purpose retrieval settings. Our experiments use public benchmarks and public implementations when available, but exact reproduction may still depend on proprietary APIs and evolving serving behavior of commercial models.

References

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

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 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.

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

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. 2025. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *Preprint*, arXiv:2508.06600.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, and 245 others. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Zirui Guo, Xiaohua Lian, Yanhua Yang, Hanzhi Huang, Shuwen Liu, Yixuan Feng, Yiding Liu, and Jinhao Li. 2024. LightRAG: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From rag to memory: Non-parametric continual learning for large language models. *Preprint*, arXiv:2502.14802.

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

Haoyu Huang, Yongfeng Huang, Junjie Yang, Zhenyu Pan, Yongqiang Chen, Kaili Ma, Hongzhi Chen, and James Cheng. 2025. Retrieval-augmented generation with hierarchical knowledge. *arXiv preprint arXiv:2503.10150*.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Li Jiapeng, Liu Runze, Li Yabo, Zhou Tong, Li Mingling, and Chen Xiang. 2024. Tree of reviews: A tree-based dynamic iterative retrieval framework for multi-hop question answering. *Preprint*, arXiv:2404.14464.

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

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *Companion Proceedings of the ACM on Web Conference 2025*.

Greg Kamradt. 2023. Needle in a haystack - pressure testing LLMs. `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`. Accessed: 2024-01-16.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. 2023. Making large language models a better foundation for dense retrieval. *Preprint*, arXiv:2312.15503.

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

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. *Preprint*, arXiv:2210.03350.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. In *Transactions of the Association for Computational Linguistics*, volume 11, pages 1316–1331.

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

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. REPLUG: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 10014–10037.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Kunlun Zhu, Yifan Luo, Dingling Xu, Yukun Yan, Zhenghao Liu, Shi Yu, Ruobing Wang, Shuo Wang, Yishan Li, Nan Zhang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. RAGEval: Scenario specific

RAG evaluation dataset generation framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8520–8544, Vienna, Austria. Association for Computational Linguistics.

Ziyuan Zhuang, Zhiyang Zhang, Sitao Cheng, Fangkai Yang, Jia Liu, Shujian Huang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Qi Zhang. 2024. Efficientrag: Efficient retriever for multi-hop question answering. *Preprint*, arXiv:2408.04259.

A Additional Methods, Experimental Details, and Supplementary Results

This appendix gathers supporting material moved out of the main paper for space, including fuller method details, supplementary results, and additional analyses that clarify the operating behavior of BEAR.

A.1 Neither-Case Failure Analysis

Among the 27 cases where neither path retrieves the supporting evidence, we find 44 missed nodes in total. All missed nodes are leaf nodes, with 93% occurring at depth 3, and 93% of the cases come from the medical domain. They are dominated by Multi-hop Reasoning and Comparison queries, indicating that these failures concentrate in settings that require cross-branch evidence collection within structurally homogeneous document types.

LLM-guided path failures. The dominant pattern is that the LLM selects the correct document but the wrong tree node (43/44 missed nodes). The main failure mode is a *conclusion-node selection bias*: during in-document node selection, the LLM tends to choose the non-leaf node whose semantics are closest to the query, often a conclusion or summary node. Such a node may summarize the topic correctly, but its subtree does not cover all of the evidence-bearing sibling branches needed for the answer. As a result, when the answer depends on evidence distributed across parallel branches such as chief complaint, physical examination, and auxiliary examinations, selecting a single summary subtree inevitably misses relevant leaf nodes in the other branches. A smaller failure mode appears in multi-document comparison queries, where the LLM sometimes selects the subtree only from the document with the stronger semantic match and misses relevant leaves in the other document.

TreeExpansion failures. TreeExpansion misses arise from two sources. The first is *entity-sparse retrieval failure*: when a query is distinguished mainly by proper nouns such as hospital or patient names, generic semantic embeddings may fail to rank the correct document highly enough within a highly homogeneous medical corpus, so the correct document falls outside the top-k. The second is *path truncation under the fixed evidence budget*. In some cases, TreeExpansion hits a relevant node in the correct document, but expansion remains confined to one subtree or sibling region, and the 4K token evidence budget is insufficient to cover the deep leaf nodes in other evidence-bearing branches. This is particularly severe in flatter document structures, where multiple relevant branches sit in parallel rather than under a single narrow subtree.

A.2 Additional Method Details

A.2.1 Qwen3-32B Full-Context Control on DragonBall

Table 2 shows that simply exposing the same Qwen3-32B generator to the full document set substantially degrades answer quality relative to structured evidence allocation. Despite receiving more raw context, the full-context control exhibits much lower Completeness and much higher Hallucination and Irrelevance, indicating that increasing available context alone is insufficient. This comparison supports the claim that BEAR's gains arise from selective evidence organization and allocation rather than from the answer model alone.

A.2.2 Hierarchical Evidence Organization and Multi-Granularity Indexing

This subsection expands the description of BEAR's offline indexing pipeline. Figure 5 summarizes the fuller offline procedure: (1) semantic-aware chunking, (2) LLM-based hierarchical tree generation, and (3) vector indexing of both internal and leaf nodes. For reproducibility, we also include the two core offline prompts used for semantic chunking and tree construction in Appendix Procedure Boxes 6 and 7, as well as the two query-time prompts used for coarse- and fine-grained LLM-guided evidence selection in Appendix Procedure Boxes 8 and 9.

Given a document collection $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, BEAR constructs a semantic forest (one tree per document) $\mathcal{F} = \{T_1, T_2, \ldots, T_N\}$, where each tree T_i is a hierarchical abstraction of document d_i. In the fuller formulation presented here, each document tree is defined as a rooted

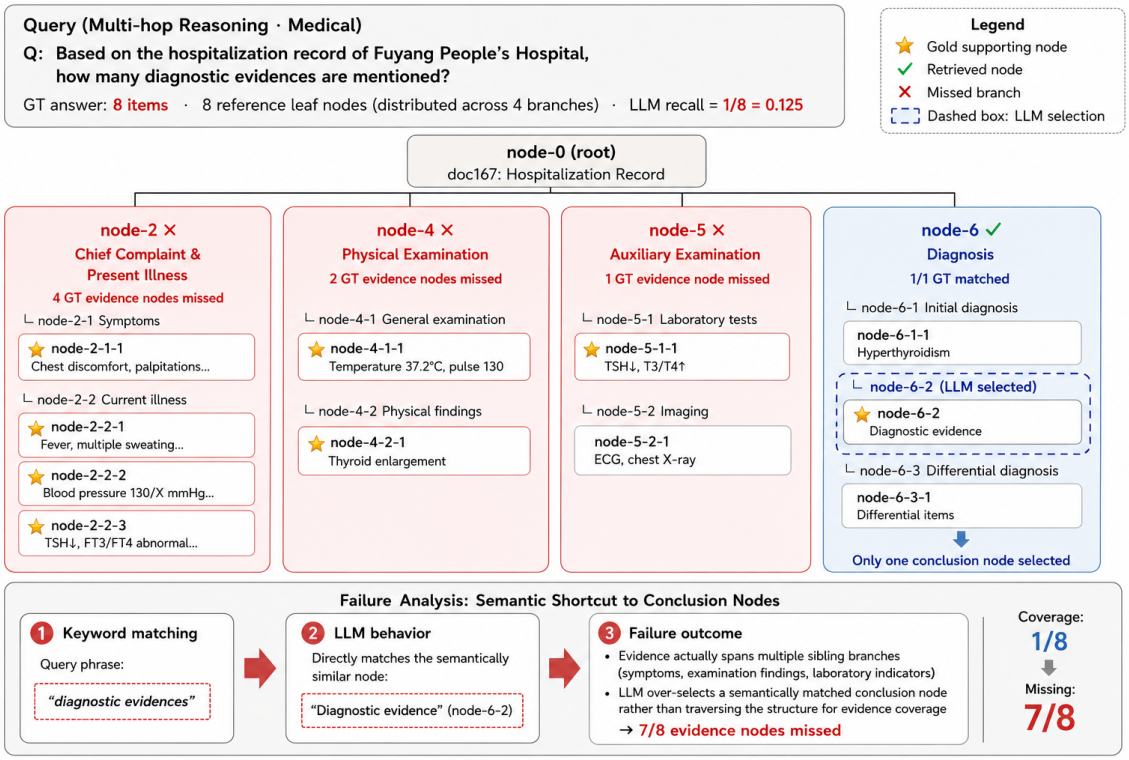

Figure 4: Failure case illustrating semantic shortcut behavior in LLM-guided node selection. The query asks for diagnostic evidence items, but the LLM directly selects a semantically matched conclusion node ("Diagnostic evidence") instead of retrieving evidence-bearing nodes distributed across multiple sibling branches. As a result, only 1 of 8 gold supporting nodes is covered.

tree $T_i = (V_i, E_i)$ with bounded depth and typed nodes (e.g., root, internal summary, and leaf content nodes). Internal nodes capture higher-level semantic abstractions, while leaf nodes retain original semantic chunks.

For each document d_i, semantic chunking produces

$$\mathcal{C}_i = \mathrm{LLM}_{\mathrm{segment}}(d_i) = \{c_1, c_2, \ldots, c_{m_i}\}, \quad (6)$$

where each chunk aligns with a semantically coherent unit rather than a fixed token window. Tree construction then maps the chunk set to a semantic hierarchy:

$$T_i = \mathrm{LLM}_{\mathrm{structure}}(\mathcal{C}_i \mid d_i). \quad (7)$$

This process jointly generates a table-of-contents-like hierarchy and node-level summaries. The implementation also supports progressive construction for long documents by building partial trees over partitions and then merging them:

$$\mathcal{T}_i^{(k)} = \textsc{TreeBuild}(\mathrm{part}_k), \quad k = 1, \ldots, K \quad (8)$$

$$\mathcal{T}_i = \textsc{TreeMerge}(\mathcal{T}_i^{(1)}, \ldots, \mathcal{T}_i^{(K)}). \quad (9)$$

These offline structures provide the substrate for allocating evidence across abstraction levels under a fixed query-time budget. To support multi-granularity retrieval, BEAR embeds non-leaf nodes using their path-level structural context and summary,

$$\mathbf{e}_v = \mathrm{Embed}(\mathrm{toc_path}(v) \oplus \mathrm{summary}(v)), \quad (10)$$

and leaf nodes using the original chunk content,

$$\mathbf{e}_c = \mathrm{Embed}(\mathrm{content}(c)). \quad (11)$$

This separation enables both summary-level retrieval and fine-grained evidence retrieval within the same indexed structure. For long documents, the progressive partition-and-merge strategy provides a practical way to scale hierarchy construction beyond a single LLM context window while still preserving document-level semantic coherence after merging the partial trees.

Table 7 summarizes the main default settings referenced in Experimental Setup. Across the main-table retrieval-based comparisons, offline hierarchy construction and online LLM-guided document/node selection use DeepSeek-V3.2, which

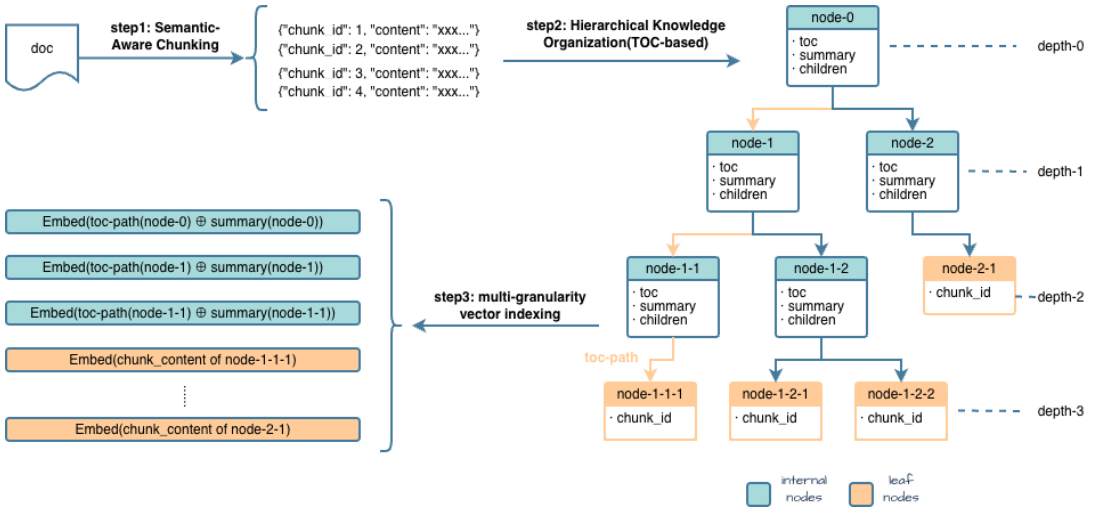

Figure 5: Hierarchical evidence organization and multi-granularity indexing for budgeted evidence access.

Procedure A0.1: Offline prompt for semantic chunking

You are a semantic text chunker.
Given the full document TEXT below, your task is to split it into a list of coherent, semantically meaningful chunks. Each chunk should represent one coherent idea or topic, such as a paragraph, a topic segment, or a semantic block.
Chunking rules:
1. *Semantic coherence:* each chunk should focus on a single topic or idea; if a new topic is introduced, start a new chunk; do not merge content discussing different concepts, even if short.
2. *Text preservation:* preserve the original text verbatim; do not paraphrase, summarize, rewrite, or reorder sentences; do not break sentences in half.
3. *Coverage and order:* every part of the original text must appear in exactly one chunk; no text may be lost or repeated; chunks must follow the original document order.
4. *Length control:* prefer chunks of roughly 100–300 words (or equivalent length); shorter chunks are acceptable if they form a complete semantic unit; avoid overly long chunks unless necessary for semantic completeness.
5. *Structure awareness:* if headings or titles appear, group them with the content they introduce; preserve paragraph boundaries and formatting where possible.
Output format: each chunk must have exactly two keys: `"id"` and `"content"`. Output must be a valid JSON array only, with no explanations, comments, markdown, or code fences.
Input document: `{document_text}`
Output the chunk list.

Figure 6: Offline prompt template used for semantic chunking. We preserve the instruction content while lightly typesetting it for appendix readability.

we access through its official API with provider-default settings, and the final answer generator is fixed to Qwen3-32B. LongRefiner is the main exception because its preprocessing relies on a separately trained helper model. To improve reproducibility and reduce implementation variance, we use public released implementations whenever available: BM25 and BGE-M3 in the main results use FlashRAG, while LongRefiner and HippoRAG2 use their official repositories. To keep retrieval comparisons controlled, non-agent pipelines align the dense embedding model to BGE-M3, and retrieval-time LLM steps use DeepSeek-V3.2 unless a baseline does not require them. BrowseComp-plus uses the same DeepSeek-V3.2 retrieval-time configuration while keeping the agent policy model fixed. In the main retrieval-based comparisons, the final evidence context passed to the shared generator is aligned to approximately 4K tokens whenever applicable. When public baseline defaults yield substantially shorter generator-side contexts, we preserve the original retrieval logic and adjust only the final evidence packing or truncation to better match the shared budget. This keeps the comparison focused on retrieval behavior while reducing variation from unequal generator-side context length. In the DragonBall node-retrieval setting used for the main ablation and mechanism analyses, each query first recalls the top 300 nodes from the FAISS index, reranks the top 30, keeps candidates with reranker score above 0.1, selects up to 5 documents after document-level grouping, and then keeps up to 5 node candidates per selected document. Other

Procedure A0.2: Offline prompt for semantic tree generation

You are a semantic document organizer.
You are given a list of pre-segmented chunks of text (`chunk_list`), where each chunk has an `"id"` and `"content"`. These chunks together represent the full content of a document, split into semantically coherent pieces. The `"id"` indicates the chunk's original order in the full document.
Your task is to analyze the entire set of chunks and produce a hierarchical JSON tree structure that reflects the document's semantic organization.
The tree must have the following levels: (1) root, representing the entire document; (2) section, a top-level thematic group; (3) sub_section, a sub-topic group; and (4) leaf nodes corresponding to the original chunk ids.
Rules:
1. Every chunk must appear once in the tree; no text may be lost; each chunk can only be assigned to one sub_section.
2. Group chunks based on semantic similarity and topic cohesiveness.
3. If a section contains chunks but no distinguishable sub_sections, group them directly under that section.
4. Use natural English labels for section and sub_section titles that reflect the topics those groups represent.
5. Output must be valid JSON with no additional explanation or text.
6. Automatically detect the language of the input document and ensure that the output summary is in the same language as the document.
Output schema: the JSON tree contains node ids, `"toc"`, `"summary"`, and `"children"`; leaf nodes additionally contain `"chunk_id"`.
Input chunk list: `{chunk_list}`
Every chunk must appear once in the tree. Generate the JSON tree now.

Figure 7: Offline prompt template used for hierarchical tree generation from the chunk list.

Procedure A0.3: Query-time prompt for document-level LLM-guided selection

You are a document retrieval expert.
You are given a user query and a list of documents.
Each document is represented by structured tree-based meta information extracted by LLM from the document.
Each document meta includes: (1) root-level TOC (document title), (2) root-level summary, and (3) level-1 child sections with TOC + summary when available. These meta fields provide a coarse but semantically rich outline of the entire document. Use this hierarchical structure to judge which documents are semantically relevant to the user's query.
User Query: `{query}`
Available Document Metas: `{doc_list_str}`
Instructions:
1. Carefully analyze the semantic meaning of the user's query.
2. For each document meta, consider all available structural fields, including root TOC, summaries, section titles, level-1 summaries, and any chunk previews.
3. Determine how likely the document can help answer the query, based on topic relevance, overlap of key entities/events/actions/dates/organizations, whether the document contains sections matching the query intent, and whether the level-1 sections indicate coverage of the required information.
4. Output must be a JSON list of document IDs only, e.g., `[12, 5, 9]`.
5. If no documents are relevant, return an empty list `[]`.
Your response (JSON list only):

Figure 8: Query-time prompt template used for document-level LLM-guided selection over document summaries and first-level section metadata.

benchmarks use the same overall retrieval pipeline but may differ in candidate counts and evidence-budget cutoffs depending on corpus scale and task format. At query time, budget control is applied after NodeFusion: we merge explicitly selected LLM nodes and structurally expanded nodes, remove redundant ancestor–descendant overlaps, prioritize LLM-selected nodes, preserve within-document order, and truncate the ordered evidence list once the target evidence budget is reached. We keep this block compact so that it clarifies the shared configuration used throughout the paper without turning the appendix into a full implementation dump.

A.2.3 Supplementary Retrieval Procedures

This subsection provides compact procedure summaries for the retrieval pipeline and NodeFusion stage. Together with the offline indexing description above, these procedures show how BEAR first performs document-level budget-aware structured retrieval and only then refines or orders evidence when finer-grained control is needed.

These two procedures make the query-time control logic more explicit. Procedure 10 shows that BEAR applies the budget check at the document level before invoking finer node-level refinement, which avoids unnecessary processing when coarse evidence already fits the evidence-context budget. Within that node-level stage, Tree-

Procedure A0.4: Query-time prompt for node-level LLM-guided selection

You are an information retrieval expert.
You are given a user query and a list of node descriptions.
Each node description is represented by structured tree-based meta information extracted by LLM from the document.
Each node meta includes: (1) document id, (2) document title, and (3) structured node information consisting of the TOC + summary fields for the corresponding section level. These meta fields provide a coarse but semantically rich outline of the candidate evidence. Use this hierarchical structure to judge which nodes are semantically relevant to the user's query.
User Query: `{query}`
Available Node Metas: `{nodes_list_info}`
Instructions:
1. Carefully analyze the semantic meaning of the user's query.
2. For each node meta, consider all available structural fields, including document title, TOC path, summaries, section titles, and any chunk previews.
3. Determine how likely the node can help answer the query, based on topic relevance, overlap of key entities/events/actions/dates/organizations, whether the node matches the query intent, and whether its local semantic context suggests useful evidence.
4. Output must be a JSON list of node IDs only, e.g., `["170_node-10", "170_node-5"]`.
5. If no nodes are relevant, return an empty list `[]`.
Your response (JSON list only):

Figure 9: Query-time prompt template used for node-level LLM-guided selection over structured node metadata.

Table 7: Default settings used across experiments unless otherwise noted.

Parameter	Default	Note
Maximum hierarchy depth	$D = 4$	shared default
Retrieval budget	1K$-$128K	varies by condition
TreeExpansion weights	$(1/3, 1/3, 1/3)$	default main setting
Fixed-length baseline chunk size	128	baseline only
Chunking / backbone model	dataset-dependent	matches benchmark language
Online LLM-guided selection	dataset-dependent	same family as offline structuring
Dense retrieval model	BGE-M3	non-agent experiments
FAISS initial recall	top-300 nodes	DragonBall node retrieval
Reranked candidates	top-30 nodes	DragonBall node retrieval
Reranker threshold	score > 0.1	DragonBall node retrieval
Document candidates	top-5 docs	DragonBall node retrieval
Node candidates per doc	top-5 nodes	DragonBall node retrieval
Retrieval-based generator	Qwen3-32B	shared answer model
DeepSeek inference config	provider default	official API setting
GPT-OSS / Qwen decoding	model-card default	HuggingFace recommended setup
NodeFusion budget rule	ordered truncation	stop at target token budget
Budget points in main curves	1K, 2K, 4K, 8K	main budget sweep

Expansion complements direct query–node similarity with ancestor-inherited and child-aggregated structural signals, allowing the retriever to exploit semantic hierarchy rather than relying only on flat embedding-space matching. Procedure 11 then shows how NodeFusion performs structure-aware deduplication, priority-based partitioning, and position-preserving ordering so that explicitly selected evidence appears before structurally expanded context while the final prompt remains both compact and readable for the generator.

A.3 Agent Benchmark Details and Results

BrowseComp-plus tests whether the retrieval gains of BEAR transfer beyond benchmark-style QA into a more realistic agent setting. These tasks require multi-document reasoning, selective evidence acquisition, and iterative access to large document collections, making the benchmark a useful downstream stress test of whether the retriever remains effective when embedded inside a broader reasoning system.

Compared with the official leaderboard entries using the same underlying retriever family, replacing the retriever with BEAR substantially improves both accuracy and recall without changing the agent policy model. This makes the transfer result easier to attribute to retrieval quality rather than to changes in agent reasoning. More broadly, the leaderboard context also shows that agent performance depends jointly on the policy model and the retriever: stronger LLMs tend to help, but retriever quality can still be a major bottleneck. In that sense, the improvement from Tongyi-

Procedure A1: Extended Budget-Aware Structured Retrieval
Input: Query q, Semantic Forest $\mathcal{F}$, hierarchy threshold L, budget $B_{\max}$
Output: Retrieved content $\mathcal{C}$
1. Collect all non-leaf nodes with depth $\leq L$ and expose their $(\mathrm{toc}, \mathrm{summary})$ pairs to the LLM.
2. Obtain document candidates $\mathcal{D}_{\mathrm{llm}}$ from LLM-guided selection.
3. Retrieve dense candidates $\mathcal{D}_{\mathrm{vector}}$ from the node index using FAISS.
4. Fuse and deduplicate the two candidate sets to form $\mathcal{D}_{\mathrm{fusion}}$.
5. If the total size of the fused document contents is within $B_{\max}$, return them directly.
6. Otherwise, perform node-level retrieval with (a) LLM-guided hierarchical navigation and (b) TreeExpansion.
7. Merge the selected nodes, order them, and apply budget control to produce the final context.

Figure 10: Appendix procedure summary for the budget-aware structured retrieval process.

Procedure A2: NodeFusion
Input: $\mathcal{N}_{\mathrm{llm}}, \mathcal{N}_{\mathrm{treexp}}$
Output: Ordered chunks $\mathcal{C}_{\mathrm{ordered}}$
1. Remove redundant ancestor–descendant overlaps from $\mathcal{N}_{\mathrm{llm}} \cup \mathcal{N}_{\mathrm{treexp}}$.
2. Partition the remaining nodes into those directly selected by the LLM and those introduced by TreeExpansion.
3. Order documents by LLM-priority first, then append TreeExpansion-only documents.
4. Within each document, sort selected nodes by their original position.
5. Convert the ordered node sequence to chunks for the generator.

Figure 11: Appendix procedure summary for the NodeFusion stage.

DeepResearch-30B-A3B/Qwen3-Embedding-8B to Tongyi-DeepResearch-30B-A3B/**BEAR(nodes)** is informative because it isolates the effect of replacing the retriever while leaving the surrounding agent backbone unchanged. The SearchCalls column is also informative: **BEAR(nodes)** matches the best reported search-call count in the table while substantially improving over the original Tongyi-DeepResearch-30B-A3B/Qwen3-Embedding-8B system on both accuracy and recall, suggesting that better retrieval quality need not require more agent search steps.

A.4 Additional Bi-Path Analysis

Table 9 reports the cross-domain and cross-language breakdown. Consistent with the main-text query-type analysis, the node-level bi-path retriever remains stronger than either single path alone across domains and languages. The relative strengths of the two single-path variants vary by slice: the TreeExpansion path is stronger in En-Finance, En-Law, and Zh-Law, whereas the LLM-guided path is stronger in En-Medical and Zh-Medical, with the two paths tied on Zh-Finance. This matches the broader pattern already visible in the query-type analysis: LLM-guided retrieval is more helpful for broad semantic synthesis, whereas the TreeExpansion path is more useful when localized evidence recovery matters, and fusion benefits from combining both signals. The gain over the stronger single path ranges from +0.00 to +13.10 points, with an average improvement of +3.80 points, showing that the complementarity between the LLM-guided path and the TreeExpansion path is not confined to a single topic or language. The largest gains appear in the more challenging Zh-Medical and En-Medical slices, suggesting that fusion is especially helpful when the retriever must combine localized evidence recovery with broader semantic guidance under more difficult conditions.

Table 11 reports a progressive node-retrieval ablation by query type, moving from flat leaf-only embedding retrieval to tree-aware retrieval and then to the stronger single-path and bi-path variants. Appendix Table 13 further shows that swapping the offline and online LLMs mainly changes the relative strength of the single-path components, especially the online LLM-guided path, while the final bi-path retriever remains comparatively stable.

The progressive ablation shows a clear staged pattern. The largest gain comes from moving from

Table 8: Performance on BrowseComp-plus. Top rows are official leaderboard results; the final row replaces the original retriever with BEAR while keeping the agent backbone fixed.

LLM	Retriever	Rank	Acc(%) ↑	Recall(%) ↑	SearchCalls ↓
GPT5	MixedbreadSearch	1st	**78.41**	48.85	44.67
GPT5	Qwen3-Embedding-8B	3rd	71.69	**78.98**	**21.74**
o3	Qwen3-Embedding-8B	4th	65.90	73.24	23.97
GPT5	BM25	5th	57.59	61.70	23.23
Tongyi-DeepResearch-30B-A3B	Qwen3-Embedding-8B	11th	44.46	62.32	30.37
Tongyi-DeepResearch-30B-A3B	**BEAR(nodes)**	–	66.60	76.60	**21.74**

Table 9: Node-level domain-language analysis on DragonBall. The Gain column reports the improvement of bi-path fusion over the stronger single path in each slice. Higher is better.

Domain-language	LLM-guided (%)	TreeExpansion (%)	Bi-path fusion (%)	Gain
En-Finance	87.18	94.79	**95.35**	+0.56 pp
Zh-Finance	94.42	100.00	**100.00**	+0.00 pp
En-Medical	91.67	86.17	**97.62**	+5.95 pp
Zh-Medical	81.17	72.26	**94.27**	+13.10 pp
En-Law	91.13	98.33	**99.17**	+0.84 pp
Zh-Law	88.80	97.22	**99.58**	+2.36 pp

pure leaf-only embedding retrieval to tree-aware retrieval over hierarchical nodes, indicating that access to internal semantic structure is an important driver of improvement. Building on that tree-aware base, both TreeExpansion and LLM-guided retrieval provide further gains, with their relative strengths varying somewhat by query type. The full bi-path retriever performs best in this appendix comparison, consistent with the complementarity claim in the main text and suggesting that hierarchical indexing and query-time bi-path access are strongest when combined.

Table 13 shows three patterns. First, the original DeepSeek-V3.2/DeepSeek-V3.2 setting remains strongest overall for the final bi-path retriever. Second, the online LLM-guided path is the most sensitive component: replacing the online selector with GPT-OSS-120B substantially reduces the standalone recall of *Tree + LLM-guided*, whereas *Leaf-only*, *Tree-aware*, and *Tree + TreeExp.* are much less affected. Third, the final **BEAR(nodes)** retriever remains comparatively stable under all four settings, with recall varying only from 96.84% to 98.19%. This suggests that the main DragonBall gains are not explained simply by using a stronger LLM, but by the interaction between consistent semantic routing and complementary structural evidence recovery.

Heuristic sensitivity. Table 10 reports the sensitivity of the TreeExpansion scoring function to different normalized mixing weights over direct similarity, ancestor inheritance, and child aggregation. The default main-paper setting uses equal weights, i.e., $\alpha = \beta = \gamma = 1/3$. The query-type breakdown reveals interpretable roles for the three signals. Increasing the direct-similarity weight tends to help multi-document questions such as information integration and temporal sequence, consistent with the need to align semantically related evidence across documents. Stronger child aggregation is most helpful for factual questions, suggesting that bottom-up support from localized descendants is especially useful when evidence is concentrated in specific subtrees. By contrast, larger inheritance weights do not yield comparable gains and can reduce overall recall, indicating that ancestor propagation alone is a weaker discriminative signal than direct matching or child-supported evidence. The balanced default remains a stable compromise even though some query types favor more specialized mixtures.

Table 12 provides a more direct view of this mechanism by attributing node-level gold-supporting evidence to the LLM-guided path, TreeExpansion, or both across query types.

The table shows that most supported evidence is shared across the two paths, but both also make nonzero unique contributions. TreeExpansion contributes the larger unique share overall (10.3% vs. 6.1%), while the LLM-guided path still recovers cases that the TreeExpansion path misses. This supports a more precise interpretation of bi-path

Table 10: TreeExpansion weight sensitivity by query type using normalized (α, β, γ) settings under a 2K budget . The default main-paper setting is $\alpha = \beta = \gamma = \frac{1}{3}$. Higher is better.

Query Type	$\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$	$0.5, 0.25, 0.25$	$0.75, 0.25, 0$	$0.25, 0.5, 0.25$	$0.25, 0.25, 0.5$	$0, 0.25, 0.75$
Recall (%)						
Factual	94.17	93.75	93.75	92.92	95.42	**95.42**
Info Integration	74.07	76.65	79.07	71.32	77.74	**80.03**
Temporal Sequence	75.84	80.97	**82.22**	71.11	76.95	76.39
Multi-hop Reasoning	**84.01**	83.68	83.68	83.48	83.42	82.92
Overall	70.93	72.44	**73.19**	68.14	71.56	71.55

Table 11: Progressive node-retrieval ablation by query type on DragonBall. We compare pure leaf-only embedding retrieval, tree-aware embedding retrieval over hierarchical nodes, TreeExpansion, LLM-guided retrieval, and the final bi-path retriever under a 4K budget where applicable. Higher is better.

Query Type	Emb (leaf-only)	Emb (Tree)	TreeExpansion	LLM-guided	Bipath (%)
Factual	72.50	92.50	94.17	86.53	**99.58**
Info Integration	62.00	92.69	92.17	89.92	**99.39**
Comparison	46.11	82.64	88.82	88.47	**96.88**
Temporal Sequence	63.96	85.28	97.22	91.94	**99.58**
Multi-hop Reasoning	34.35	84.22	86.32	86.67	**95.26**
Summarization	24.40	90.13	90.62	91.99	**98.47**
Overall	50.55	87.91	91.55	89.25	**98.19**

retrieval: its benefit comes from a strong shared retrieval core plus asymmetric partial complementarity, rather than from two fully redundant or perfectly symmetric mechanisms. Multi-hop Reasoning is the clearest difficult case, with the lowest overlap (72.3%) and the highest uncovered fraction (8.4%), which helps explain why combining the two paths is especially valuable when evidence must be assembled across multiple reasoning steps.

Table 12: Gold-supporting-evidence attribution by query type at the node level. Bi-path Coverage denotes the union of the two paths, while Neither indicates the fraction of gold-supporting evidence recovered by neither path.

Query Type	Only LLM-guided (%)	Both (%)	Only TreeExpansion (%)	Bi-path (%)	Neither (%)
Factual	5.3%	77.9%	16.0%	99.2%	0.8%
Info Integration	7.1%	82.7%	9.0%	98.8%	1.2%
Comparison	7.4%	79.1%	10.1%	96.5%	3.5%
Temporal Sequence	2.5%	88.8%	8.3%	99.6%	0.4%
Multi-hop Reasoning	6.7%	72.3%	12.6%	91.6%	8.4%
Summarization	6.7%	82.9%	8.6%	98.2%	1.8%
Overall	6.1%	80.7%	10.3%	97.3%	2.9%

Table 13: Sensitivity of DragonBall node retrieval to the offline and online LLM choice. We swap the LLM used for offline hierarchy construction and the LLM used for online LLM-guided selection between DeepSeek-V3.2 and GPT-OSS-120B, while keeping the remaining pipeline unchanged. Row names are aligned with the progressive node-retrieval ablation in the main text. Higher is better.

Variant	offline:dsv32 online:dsv32	offline:dsv32 online:oss120b	offline:oss120b online:dsv32	offline:oss120b online:oss120b
Recall (%)				
Leaf-only	50.55	51.11	53.64	53.64
Tree-aware	87.91	89.26	86.21	86.21
Tree + LLM-guided	89.25	71.19	79.91	69.46
Tree + TreeExp.	91.55	94.31	93.25	93.25
BEAR(nodes)	98.19	96.84	97.30	96.96