Per-Level Routing Error Compounds Exponentially: A Negative Result on Embedding-Based Hierarchical Retrieval at Scale

Jason Crispin

jason@auditsu.com

February 2026

Abstract

Hierarchical retrieval promises token-efficient context for large language model (LLM) systems by routing queries through a tree of summaries rather than searching a flat index. We present HCR (Hierarchical Context Retrieval), a coarse-to-fine retrieval system that uses embedding-based routing through LLM-generated contrastive summaries organized in a tree built by bisecting k-means clustering. We introduce **per-level routing epsilon**, a novel metric that measures routing error at each tree level independently, enabling diagnosis of where hierarchical retrieval fails. We tested 12 configurations across two corpus scales (315 and 21,897 chunks), varying beam width (3–8), embedding model (MiniLM 384-dim, mpnet 768-dim), routing strategy (cosine, cross-encoder, BM25 hybrid), and summary representation. At small scale, the best configuration achieved nDCG@10 = 0.580 versus a flat dense retrieval + cross-encoder baseline of 0.835. At medium scale, HCR collapsed to nDCG@10 = 0.094 versus 0.749 for the baseline—a catastrophic failure triggered by exponential error compounding across tree depth. Per-level epsilon analysis confirms the mechanism: routing error at each level compounds as $\prod(1 - \varepsilon_l)$, and at depth 5, even moderate per-level error produces near-random leaf selection. Beam width sweep confirms that the tree structure is sound (correct branches exist) but embedding-based cosine routing cannot reliably select them. We document five empirical patterns—DPI summary blindness, cross-encoder routing damage, beam width as diagnostic, embedding saturation, and BM25 routing sparsity—that inform future hierarchical retrieval design. We argue that per-level epsilon should be a standard evaluation metric for any hierarchical retrieval system.

1 Introduction

The growth of LLM context windows has not eliminated the need for precise retrieval. Larger windows enable longer inputs but do not guarantee that the model attends to the right information. For agentic systems operating under token budgets—where every token of context costs inference time and money—retrieval quality matters more than retrieval quantity. The goal is *minimum viable context*: the fewest tokens that produce a correct answer.

Hierarchical retrieval offers an appealing path to this goal. By organizing documents into a tree of increasingly specific summaries, a system can route queries coarse-to-fine, pruning irrelevant branches early and delivering only the most relevant leaf content. Systems like RAPTOR [1] and LATTICE [2] have demonstrated that tree-structured indices improve performance on complex reasoning tasks. The theoretical basis is straightforward: a tree with branching factor b and depth d reduces the search space from N to $O(b \cdot d)$ scoring operations, with proportional token savings.

But this efficiency depends on routing accuracy. The Data Processing Inequality guarantees that information is lost ascending the tree—summaries are lossy compressions of their children [3]. If the routing mechanism makes errors, those errors compound multiplicatively across levels. At depth d with per-level error rate ε, end-to-end recall degrades as $(1-\varepsilon)^d$. This compounding is structural: it applies to any system that makes sequential, irrevocable routing decisions through a hierarchy.

We built HCR to test whether embedding-based routing through LLM-generated summaries could achieve sufficient accuracy to make hierarchical retrieval practical under token constraints. We tested 12 configurations systematically, varying every major design parameter. The result is a clear negative: **embedding-based tree routing fails at scale due to exponential error compounding.** At 315 chunks, the best configuration reached nDCG@10 = 0.580. At 21,897 chunks (a 70× increase), performance collapsed to 0.094—worse than random retrieval with the token budget.

This paper makes three contributions:

1. **Per-level routing epsilon (ε):** A novel metric that measures routing accuracy at each tree level independently. No prior system has exposed or measured per-level routing decisions. This metric reveals *where* hierarchical retrieval fails, not just *that* it fails.

2. **A systematic negative result:** Twelve configurations tested across two scales, two embedding models, four routing strategies, and three summary representations. The failure is not due to a single bad design choice—it is structural to embedding-based routing on tree-organized summaries.

3. **Five empirical patterns** that explain specific failure mechanisms and inform the design of future hierarchical retrieval systems.

2 Related Work

2.1 Hierarchical Retrieval Systems

RAPTOR [1] introduced tree-structured retrieval for LLMs, using bottom-up clustering with LLM-generated summaries at each level. Critically, RAPTOR's own experiments show that its *collapsed tree* mode (flat retrieval over all nodes at all levels) outperforms strict top-down tree traversal. This finding is theoretically predicted by error compounding analysis [3] and is a key motivation for our investigation.

LATTICE [2] (UT Austin, October 2025) is the closest system to HCR. It uses LLM-guided tree navigation with logarithmic complexity, sharing the philosophy of hierarchical routing through summaries. LATTICE differs in using LLM-as-judge scoring at each level (expensive but potentially more accurate) and sequential best-first traversal rather than parallel beam search. Neither RAPTOR nor LATTICE report per-level routing accuracy.

HIRO [4] implements DFS-style querying on RAPTOR trees with delta-threshold pruning, the closest prior work to elimination-based traversal. GraphRAG [5] uses hierarchical community detection in a graph structure rather than a tree, at substantially higher cost.

2.2 Flat Retrieval Baselines

Modern flat retrieval combines dense embedding search with cross-encoder reranking [6]. Cross-encoders (e.g., MS-MARCO trained models) jointly encode query-document pairs, achieving substantially higher accuracy than bi-encoder similarity at the cost of $O(N)$ inference. For corpora under ∼100K chunks, flat + cross-encoder retrieval is practical and provides a strong baseline.

2.3 Theoretical Foundations

The fragility of hierarchical elimination is well-established in information theory. The Data Processing Inequality [3] guarantees that mutual information between a query and a node's content cannot increase as one ascends the tree. Branch-and-bound optimality requires admissible bounds on descendant relevance—a condition that embedding similarity to summary text does not satisfy. The cluster hypothesis [7] (that relevant documents cluster together) is necessary for tree-based retrieval but fails in 46% of broad collections [8].

Three independent analyses converge on the same formula: per-level miss rate ε compounds as $(1-\varepsilon)^d$, making deep trees with imperfect routing exponentially fragile. Beam search mitigates this by maintaining k candidates, transforming recall to approximately $(1-(1-p)^k)^d$, but cannot eliminate the fundamental compounding.

2.4 The Measurement Gap

No prior system has measured per-level routing accuracy. RAPTOR, LATTICE, HIRO, and other hierarchical systems report end-to-end metrics (nDCG, recall, F1) but do not expose the internal routing decisions that determine whether the correct branches are traversed at each level. This makes it impossible to diagnose *where* hierarchical retrieval fails. Our per-level epsilon metric fills this gap.

3 Method

3.1 Tree Construction

HCR builds a retrieval tree through a two-stage process: hierarchical clustering followed by LLM-based summary generation.

Clustering. We use top-down bisecting k-means in embedding space. Starting from the full corpus, each level splits into b clusters (target branching factor), recursing until maximum depth d is reached or clusters are small enough to serve as leaves. For the small corpus (315 chunks), the tree has depth 3 with branching factor 8, producing a structure: L0: 1 root, L1: 8 nodes, L2: 64 nodes, L3–L4: 333 leaves. For the medium corpus (21,897 chunks), depth increases to 5 with 25,716 total nodes (3,819 internal).

Summary generation. For each internal node, an LLM generates a structured routing summary containing:

- **theme**: cluster topic (1–2 sentences)
- **includes**: 5–8 specific topics covered
- **excludes**: 3–5 topics NOT covered (contrastive, informed by sibling nodes)
- **key_entities**: 5–10 proper nouns and product names
- **key_terms**: 8–15 searchable keywords, abbreviations, synonyms

The contrastive design (explicitly stating what a node does *not* cover, based on its siblings) is intended to improve discriminability between sibling nodes. Summaries are concatenated into a single text string for embedding.

Embedding. Summary text is embedded using sentence-transformers models. We tested two models: all-MiniLM-L6-v2 (384 dimensions) and all-mpnet-base-v2 (768 dimensions). All embeddings are L2-normalized. When changing embedding models, the tree must be rebuilt—using a different model's embeddings with a tree clustered in a different embedding space degrades performance (Section 5.1).

3.2 Query-Time Routing

At query time, HCR performs beam search traversal through the tree.

Scoring cascade. At each level, all children of the current frontier nodes are scored against the query. For internal nodes, scoring uses cosine similarity between the query embedding and the node's summary embedding. For leaf nodes, a cross-encoder (cross-encoder/ms-marco-MiniLM-L-6-v2) reranks the top candidates by jointly encoding query-chunk pairs.

Cross-encoder scoring is *disabled* for internal nodes. We empirically determined that the MS-MARCO cross-encoder, trained on natural language query-passage pairs, performs poorly on structured routing summary text (Section 6.2, Pattern 2).

Beam selection. At each level, the top k candidates (beam width) are retained. Path-relevance EMA smooths scores across depth: $s_{\text{new}} = \alpha \cdot s_{\text{current}} + (1 - \alpha) \cdot s_{\text{parent}}$, with $\alpha = 0.5$. An MMR-style diversity penalty discourages multiple beams from exploring the same branch ($\lambda_{\text{diversity}} = 0.3$).

Token budget. Retrieved leaf chunks are packed greedily up to a 400-token budget. We use relevance-minus-redundancy scoring for chunk selection within the budget.

3.3 Per-Level Routing Epsilon (ε)—Novel Metric

We introduce per-level routing epsilon to measure routing accuracy at each tree level independently.

Definition. For a set of evaluation queries Q with known relevant chunks, and a tree with levels $l = 0, 1, \ldots, L$:

$$\varepsilon_l = 1 - \frac{|\{q \in Q : \text{ancestor}_l(\text{gold}(q)) \in \text{beam}_l(q)\}|}{|Q|} \tag{1}$$

where $\text{ancestor}_l(\text{gold}(q))$ is the ancestor at level l of any gold-relevant chunk for query q, and $\text{beam}_l(q)$ is the set of nodes in the beam at level l for query q. A query is "correct" at level l if *any* of its gold chunks' ancestors appear in the beam.

Interpretation. $\varepsilon_l = 0$: perfect routing at level l (all relevant branches in beam). $\varepsilon_l = 1$: random routing (no relevant branches in beam).

Theoretical prediction. If routing errors are independent across levels, end-to-end recall is approximately:

$$R \approx \prod_{l=1}^{L} (1 - \varepsilon_l) \tag{2}$$

At depth 5 with $\varepsilon = 0.15$ per level: $R \approx 0.85^5 = 0.444$. At $\varepsilon = 0.30$: $R \approx 0.70^5 = 0.168$.

Why this hasn't been measured. Prior hierarchical retrieval systems (RAPTOR, LATTICE, HIRO) report only end-to-end metrics. They do not instrument or expose the beam contents at each intermediate level. Per-level epsilon requires (a) ground-truth relevant chunks with known positions in the tree, and (b) logging of beam state at every tree level during evaluation. HCR instruments both.

4 Experimental Setup

4.1 Corpus

We evaluate at two scales using the GitLab public handbook as source material:

Table 1: Corpus statistics

	Small	**Medium**
Documents	50	3,885
Chunks	315	21,897
Total tokens	125,435	8,804,259
Mean chunk tokens	398	402
Source	GitLab handbook (sample)	GitLab handbook (full)

The GitLab handbook is a large, publicly available organizational knowledge base covering company processes, policies, engineering practices, and operational procedures. It provides realistic heterogeneity (technical documentation, HR policies, financial processes, engineering guides) representative of organizational retrieval tasks.

4.2 Queries

Small: 50 queries stratified across 9 categories: single_branch (12), entity_spanning (10), dpi (7), multi_hop (5), comparative (5), aggregation (2), temporal (2), ambiguous (2), ood (5).

Medium: 150 queries with proportional stratification across the same categories.

Queries were generated by an LLM from source chunks, with category labels reflecting the structural retrieval challenge each query poses (e.g., "dpi" queries target specific details that summaries may lose; "comparative" queries require information from multiple branches).

4.3 Baselines

Three baselines of increasing sophistication:

1. **BM25:** Okapi BM25 with $k_1 = 1.5$, $b = 0.75$. Lexical baseline.
2. **Hybrid-RRF:** BM25 + dense retrieval combined via reciprocal rank fusion. Standard hybrid approach.
3. **Flat+CE:** Dense retrieval followed by cross-encoder reranking (cross-encoder/ms-marco-MiniLM-L-6-v2). This is the **kill baseline**—HCR must exceed this to validate the hierarchical approach.

4.4 HCR Configurations

We tested 12 configurations (v1–v12), of which 11 are valid (v1 was discarded due to an implementation bug in the tree builder). Configurations are organized by the experimental question they address (Table 2).

Table 2: HCR configurations tested

Config	Variable	Beam	Embed	Routing	Key Change
v2	Baseline	3	MiniLM	CE all	First proper tree
v3	Routing	3	MiniLM	Cosine	Removed CE from routing
v4	Beam width	5	MiniLM	Cosine	Wider beam
v5	Beam ceiling	8	MiniLM	Cosine	Maximum beam
v6	Summary text	5	MiniLM	Cosine	Enriched embed text
v7	Summary text	5	MiniLM	Cosine	Contrastive prompts
v8	Summary text	5	MiniLM	Cosine	Content snippets
v9	Embed model	5	mpnet	Cosine	No tree rebuild
v10	Embed + rebuild	5	mpnet	Cosine	Rebuilt tree
v11	Ceiling	8	mpnet	Cosine	Best config, max beam
v12	Hybrid routing	5	mpnet	BM25+cos	BM25 hybrid

4.5 Metrics

Standard IR: nDCG@10 (normalized discounted cumulative gain), Recall@10 (fraction of gold chunks in top 10), MRR (mean reciprocal rank).

Token efficiency: MeanTok (average tokens used under 400-token budget).

Novel: Per-level ε (routing error at each tree level, Section 3.3); Sibling Distinctiveness (SD, mean pairwise cosine distance between sibling embeddings; values above 0.15 indicate sufficient separation).

4.6 Fail-Fast Protocol

We implement a fail-fast evaluation sequence with kill criteria at each step:

1. **Topology check:** SD $< 0.15 \Rightarrow$ KILL
2. **Routing check:** L1 $\varepsilon > 0.05$ on easy queries $\Rightarrow$ WARNING
3. **nDCG check:** HCR nDCG $<$ Flat+CE nDCG $-0.15 \Rightarrow$ KILL
4. **Token efficiency curve:** diminishing returns check
5. **Full evaluation:** all metrics

5 Results

5.1 Small-Scale Results (315 chunks)

Baselines. Table 3 shows baseline performance at small scale.

Table 3: Baseline results, small corpus (315 chunks, 50 queries)

System	nDCG@10	Recall@10	MRR	MeanTok
BM25	0.705	0.82	0.669	333
Hybrid-RRF	0.719	0.90	0.662	343
Flat+CE	**0.835**	**0.94**	**0.803**	354

Flat+CE substantially outperforms both lexical and hybrid baselines, establishing a strong kill criterion of nDCG@10 = 0.835.

Table 4: HCR results, small corpus (315 chunks, 50 queries)

Config	Key Variable	nDCG	R@10	MRR	Tok	L1 ε	L2 ε
v2	CE all, beam=3	0.318	0.40	0.298	234	0.32	0.62
v3	Cosine, beam=3	0.287	0.32	0.281	235	0.32	0.58
v4	Beam=5	0.420	0.46	0.411	255	0.24	0.46
v5	Beam=8 (MiniLM)	0.509	0.58	0.490	297	0.14	0.36
v6	Enriched text	0.493	0.54	—	249	0.16	0.36
v7	Contrastive prompts	0.484	—	—	269	0.14	0.44
v8	Content snippets	0.485	—	—	279	0.16	0.42
v9	mpnet, no rebuild	0.450	0.54	—	296	0.18	0.38
v10	mpnet, rebuilt	0.540	0.62	0.516	231	0.24	0.42
v11	mpnet, beam=8	**0.580**	**0.66**	**0.556**	290	**0.06**	**0.28**
v12	BM25 hybrid	0.465	—	—	—	—	—

HCR progression. Table 4 shows the systematic improvement across configurations. Key observations at small scale:

1. **Beam width is the strongest lever.** Increasing beam from 3 to 8 (v3→v5) improved nDCG by +0.222 and halved L1 ε from 0.32 to 0.14. This is monotonic across all beam widths tested.

2. **Cross-encoder routing is net negative.** v2 (CE at all levels) vs v3 (cosine-only) shows CE slightly *improves* nDCG (+0.031) but with worse recall (0.40 vs 0.32). Analysis revealed the MS-MARCO cross-encoder flipped 26 correct cosine decisions to incorrect while saving only 14—a net loss of 12 correct routing decisions.

3. **Summary text enrichment helps, then plateaus.** v6 (enriched) improved L1 ε from 0.24 to 0.16, but contrastive prompts (v7) and content snippets (v8) provided no additional benefit.

4. **Embedding model matters, but only with tree rebuild.** mpnet without rebuilding the tree (v9) *decreased* nDCG by 8.7% compared to MiniLM (v6). After rebuilding (v10), nDCG improved to 0.540—the best at beam=5.

5. **Best configuration (v11):** mpnet with rebuilt tree at beam=8 achieves nDCG = 0.580, L1 ε = 0.06, L2 ε = 0.28. The gap to Flat+CE remains 0.255.

Per-level epsilon analysis. Table 5 shows the full per-level epsilon for the default HCR configuration.

Table 5: Per-level epsilon, HCR default configuration (small corpus)

Level	Queries Evaluated	Correct in Beam	ε
L0 (root)	50	50	0.00
L1	50	39	0.22
L2	50	25	0.50
L3	50	21	0.58
L4 (leaves)	31	5	0.84

The monotonic degradation of routing accuracy with depth is stark. At L4, only 5 of 31 queries that reached the leaf level had correct chunks in the beam—effectively random selection. The bottleneck shifts with beam width: at beam=8 (v11), L1 ε drops to 0.06, but L2 remains at 0.28, indicating that deeper levels become the binding constraint as shallow routing improves.

5.2 Medium-Scale Results (21,897 chunks)

At medium scale, the tree grows to 25,716 nodes (3,819 internal) with depth 5 and 8 root branches. Sibling distinctiveness is 0.445 (passing the 0.15 threshold).

Table 6: Baseline results, medium corpus (21,897 chunks, 150 queries)

System	nDCG@10	Recall@10	MRR	MeanTok
BM25	0.522	0.69	0.472	336
Hybrid-RRF	0.594	0.77	0.548	352
Flat+CE	**0.749**	**0.86**	**0.717**	349

All baselines degrade from small to medium scale, as expected with a 70× increase in corpus size. Flat+CE drops from 0.835 to 0.749 but remains the strongest baseline.

HCR result. The fail-fast protocol killed HCR at step 3 (nDCG check):

Table 7: HCR fail-fast result, medium corpus

Metric	Value
Sibling Distinctiveness	0.445 (pass)
HCR nDCG@10	**0.094**
Flat+CE nDCG@10	0.749
Gap (HCR – Flat+CE)	**−0.655**
Kill threshold	−0.15
Outcome	**KILLED**

HCR nDCG collapsed from 0.580 (small, best config) to 0.094 (medium)—an 84% relative decline. The gap of −0.655 to the kill baseline is more than four times the kill threshold.

5.3 Scale-Dependent Collapse

Table 8 directly compares small and medium performance.

Table 8: Scale comparison

Metric	Small (315)	Medium (21,897)	Change
Flat+CE nDCG	0.835	0.749	−10.3%
HCR nDCG (best)	0.580	0.094	**−83.8%**
HCR MeanTok	290	292	+0.7%
Tree depth	3	5	+2 levels
Internal nodes	∼73	3,819	+52×

The key observation: **token efficiency is maintained** (290→292 tokens) while **accuracy collapses**. The tree successfully limits token usage, but the tokens it selects are wrong. The additional two levels of depth at medium scale (depth 3→5) multiply routing errors through two more compounding stages.

Using the compound error model: even if each level achieved $\varepsilon = 0.15$ (better than most observed values), five levels would yield $0.85^5 = 0.444$—less than half of queries correctly routed. The observed nDCG of 0.094 suggests actual per-level error is substantially worse at medium scale, consistent with increased difficulty of discriminating among more siblings at each level.

6 Discussion

6.1 Why Embedding-Based Routing Fails at Scale

The mechanism is $(1-\varepsilon)^d$, confirmed empirically. Three factors combine to make this fatal at scale:

Depth increases with corpus size. A tree with branching factor 8 requires depth 3 for 315 chunks but depth 5 for 21,897 chunks. Each additional level multiplies the cumulative error. There is no way to avoid this while maintaining logarithmic scoring cost.

Per-level epsilon does not improve with scale. Embedding cosine similarity between a query and a structured routing summary is a function of the summary's quality and the embedding model's capacity. Adding more chunks to the corpus makes each internal node's children more numerous and more similar, *increasing* ε at each level. The problem gets harder, not easier, with scale.

The interaction is multiplicative, not additive. A system that is "pretty good" at routing ($\varepsilon = 0.15$ per level) is catastrophic at depth 5: $0.85^5 = 0.444$ recall before any leaf-level errors. At $\varepsilon = 0.25$: $0.75^5 = 0.237$. The exponential decay is unforgiving.

This explains why the small-scale results were misleadingly optimistic. At depth 3 with the best configuration ($\varepsilon_1 = 0.06$, $\varepsilon_2 = 0.28$), the compounding is manageable. Adding two levels transforms a viable system into a non-functional one.

6.2 Five Empirical Patterns

Our systematic exploration of 12 configurations revealed five patterns that explain specific failure mechanisms.

Pattern 1: DPI Summary Blindness. Routing summaries, by design, compress information. The Data Processing Inequality guarantees they lose detail. Queries targeting specific facts (dates, version numbers, conditional logic, quantities) score poorly against thematic summaries because the discriminative features are absent. The structured summary format (key_entities, key_terms) partially mitigates this but cannot recover information that was never salient enough to include.

Pattern 2: Cross-Encoder Routing Damage. The MS-MARCO cross-encoder, trained on natural language query-passage pairs, is poorly calibrated for structured routing metadata. When applied to internal node scoring (v2), it flipped 26 correct cosine routing decisions to incorrect while correcting only 14—a net loss. Routing summaries (structured lists of topics, entities, and keywords) are distributionally different from MS-MARCO training data. Cross-encoder scoring remains effective at the leaf level, where it operates on actual chunk text.

Pattern 3: Beam Width as Diagnostic. Increasing beam width from 3 to 8 monotonically improved both nDCG and per-level epsilon across all configurations. This is diagnostic: if the tree structure were fundamentally broken (correct branches did not exist), wider beams would not help. The monotonic improvement confirms that the tree contains the right information in the right places—the routing mechanism simply cannot find it reliably.

Pattern 4: Embedding Saturation. Three approaches to improving summary embeddings (enriched text, contrastive prompts, content snippets) all plateaued at similar nDCG levels ($\sim$0.48–0.49) with MiniLM (384-dim). Switching to mpnet (768-dim) with a rebuilt tree improved nDCG to 0.540, but the gains came primarily from better *leaf-level* discrimination (L4 ε improved from 0.81 to 0.75), not from routing improvement (L1 ε actually worsened from 0.16 to 0.24). Routing quality improvement requires a different lever than embedding dimensionality.

Pattern 5: BM25 Routing Sparsity. Hybrid BM25+cosine routing via reciprocal rank fusion (v12) degraded nDCG by 14% compared to cosine-only routing (v10). BM25 operates on routing summaries containing only $\sim$10–20 tokens of keywords per node. This is too sparse

for meaningful lexical matching, producing near-random BM25 scores that inject noise into the fusion.

6.3 Implications for Hierarchical Retrieval

The tree structure is not the problem. Beam width sweep (Pattern 3) confirms that correct branches exist at every level. Sibling distinctiveness passes at both scales (0.608 small, 0.445 medium).

The routing mechanism is the problem. Embedding cosine similarity between a query and a structured routing summary is fundamentally limited in discriminative power. The summary embedding must simultaneously encode topic scope, entity coverage, keyword relevance, and contrastive exclusions in a single vector.

RAPTOR's design insight is validated. RAPTOR's collapsed-tree approach—flat retrieval over nodes at all levels, avoiding top-down routing entirely—sidesteps the error compounding problem. Our results provide a quantitative explanation for *why* collapsed tree outperforms tree traversal.

Per-level epsilon should be standard. Any hierarchical retrieval system making sequential routing decisions through a tree should report per-level epsilon alongside end-to-end metrics. Without it, there is no way to determine whether poor performance is due to tree structure, routing mechanism, or leaf-level retrieval.

7 Conclusion

We built HCR to test whether embedding-based routing through LLM-generated summaries could make hierarchical retrieval practical under token constraints. The answer is no. At 21,897 chunks, HCR achieved nDCG@10 of 0.094 versus 0.749 for flat retrieval with cross-encoder reranking—a catastrophic failure caused by exponential compounding of per-level routing error across five tree levels.

The per-level routing epsilon metric we introduce reveals exactly where and why this failure occurs. At each level, embedding cosine similarity against structured routing summaries makes moderate errors ($\varepsilon \approx 0.15$–0.30 at the best configurations). These errors compound multiplicatively with depth, producing near-random leaf selection at the scales where hierarchical retrieval is most needed.

Three points deserve emphasis:

1. **The tree structure works.** Beam width sweep confirms that correct branches exist at every level. Content is well-organized; routing cannot find it.

2. **The routing mechanism fails.** Embedding-based cosine similarity on structured summaries is not discriminative enough for reliable multi-level routing. This is a fundamental limitation, not a tuning problem—we tested four routing strategies, two embedding models, and three summary representations.

3. **Per-level epsilon is necessary.** Without per-level measurement, the small-scale nDCG of 0.580 might have appeared promising. Per-level epsilon revealed that this performance was supported by shallow depth (3 levels), not by routing accuracy, predicting the collapse at medium scale.

Future work should decouple tree construction (which works) from routing (which does not). Promising directions include learned routing functions trained specifically on tree-structured retrieval decisions, LLM-as-judge routing (as in LATTICE, though at higher cost), or hybrid architectures that use the tree for representation enrichment while avoiding strict top-down routing for final retrieval.

References

[1] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C.D. Manning. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. In *ICLR*, 2024.

[2] N. Gupta, W.-C. Chang, N. Bui, C.-J. Hsieh, and I.S. Dhillon. LLM-guided Hierarchical Retrieval. *arXiv:2510.13217*, 2025.

[3] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

[4] A. Goel and A. Chandak. HIRO: Hierarchical Information Retrieval Optimization. *arXiv:2406.09979*, 2024.

[5] Microsoft. GraphRAG: Graph-based Retrieval-Augmented Generation, 2024.

[6] R. Nogueira and K. Cho. Passage Re-ranking with BERT. *arXiv:1901.04085*, 2019.

[7] C.J. van Rijsbergen. *Information Retrieval*. Butterworths, 2nd edition, 1979.

[8] E.M. Voorhees. The cluster hypothesis revisited. In *Proceedings of SIGIR*, 1985.

A Full Configuration Table

All 12 configurations with complete metrics. v1 discarded due to tree builder bug.

Table 9: Complete HCR configuration results (small corpus, 315 chunks, 50 queries)

Cfg	Bm	Dims	Tree	Route	Text	nDCG	R@10	MRR	Tok	L1 ε	L2 ε
v1	3	384	MiniLM	CE all	Def	0.345	—	—	—	—	—
v2	3	384	MiniLM	CE all	Def	0.318	0.40	0.298	234	0.32	0.62
v3	3	384	MiniLM	Cosine	Def	0.287	0.32	0.281	235	0.32	0.58
v4	5	384	MiniLM	Cosine	Def	0.420	0.46	0.411	255	0.24	0.46
v5	8	384	MiniLM	Cosine	Def	0.509	0.58	0.490	297	0.14	0.36
v6	5	384	MiniLM	Cosine	Enr	0.493	0.54	—	249	0.16	0.36
v7	5	384	MiniLM	Cosine	Con	0.484	—	—	269	0.14	0.44
v8	5	384	MiniLM	Cosine	Snp	0.485	—	—	279	0.16	0.42
v9	5	768	MiniLM	Cosine	Enr	0.450	0.54	—	296	0.18	0.38
v10	5	768	mpnet	Cosine	Enr	0.540	0.62	0.516	231	0.24	0.42
v11	8	768	mpnet	Cosine	Enr	**0.580**	**0.66**	**0.556**	290	**0.06**	**0.28**
v12	5	768	mpnet	BM25+	Enr	0.465	—	—	—	—	—

Bm = beam width; Dims = embedding dimensions; Text: Def = default, Enr = enriched, Con = contrastive, Snp = snippets. "—" = not recorded. v1 discarded (tree builder bug). v9 uses mpnet embeddings on a MiniLM-clustered tree (misaligned).

B Per-Category Breakdown (Small Scale)

Baseline performance by category.

HCR performance by category (v12 configuration).

HCR's failure is concentrated in single_branch and comparative queries. Single_branch queries should be the easiest for hierarchical retrieval, yet HCR fails on 67% of them—indicating routing errors at early levels misdirect the beam.

Table 10: Flat+CE baseline performance by query category (small corpus)

Category	N	CE nDCG@10	CE–BM25 Gap
ambiguous	2	0.565	+0.315
dpi	7	0.615	+0.006
comparative	5	0.663	+0.451
single_branch	12	0.792	+0.023
entity_spanning	10	1.000	+0.137
multi_hop	5	1.000	+0.274

Table 11: HCR performance by query category (small corpus, v12)

Category	N	nDCG@10	R@10	Notes
single_branch	12	0.244	0.33	8/12 scored 0.0
entity_spanning	10	0.663	0.70	Best HCR category
dpi	7	0.574	0.71	Bimodal
multi_hop	5	0.400	0.40	2/5 perfect, 3/5 zero
comparative	5	0.077	0.20	Near-complete failure
aggregation	2	0.500	0.50	N too small
temporal	2	1.000	1.00	Perfect (N=2)
ambiguous	2	0.816	1.00	Strong (N=2)
ood	5	0.526	0.60	Mixed

C Routing Diagnosis: Cosine vs Cross-Encoder

Configuration v2 applied the MS-MARCO cross-encoder at all tree levels. Configuration v3 used cosine similarity only.

Table 12: Cosine vs cross-encoder routing comparison

Metric	v2 (CE all)	v3 (Cosine)	Delta
nDCG@10	0.318	0.287	+0.031
Recall@10	0.40	0.32	+0.08
L1 ε	0.32	0.32	0.00
L2 ε	0.62	0.58	+0.04

Detailed analysis of L1 routing decisions revealed 26 queries where cosine was correct but CE flipped to incorrect, and 14 where CE corrected cosine—a net loss of 12 correct decisions. The MS-MARCO cross-encoder assigns unreliable scores to structured routing metadata, which is distributionally different from its training data (natural language query-passage pairs). Example routing summary format:

```
Theme: GitLab procurement and vendor management
Covers: purchase orders, vendor onboarding, Coupa system
Not: engineering practices, HR policies, product development
Entities: Coupa, ZIP, Navan, GitLab Procurement
Terms: requisition, PO, vendor, procurement, contract
```

Design implication: Cross-encoder reranking should be restricted to leaf-level scoring, where it operates on natural language chunk text matching its training distribution.