Semantic Compression Trees: Multi-Resolution Knowledge Retrieval via Hierarchical Semantic Residuals

Junaid Farooq
Sprouts.ai
junaid.farooq@sprouts.ai
National Institute of Technology Srinagar
junaid_phd017@nitsri.ac.in
junaidfarooq.net

August 25, 2026

Abstract

Retrieval-augmented generation relies mostly on flat, fixed-granularity indexes: documents are cut into uniform chunks and retrieved by similarity, discarding the hierarchical structure of the source. We introduce **Semantic Compression Trees (SCT)**, a hierarchical index in which each node stores only its *semantic residual* — the information it adds beyond its parent — and retrieval proceeds by progressive descent from the root, so that per-query cost is governed by tree depth rather than collection size.

We evaluate on QASPER [Dasigi et al., 2021] (50 papers, 173 questions) under two protocols differing only in whether the benchmark supplies the relevant document, with bootstrap confidence intervals and paired significance tests throughout. The results are mixed and we report them as such. When the document is given, SCT with a zero-LLM extractive compressor matches dense retrieval on answer quality (0.274 vs. 0.277 F1, $p = 0.37$) using **30% fewer context tokens** and **no LLM calls** to build the index, and residual storage beats storing full summaries at each node (0.274 vs. 0.205, $p < 0.001$). Increasing the collection fifty-fold multiplies flat retrieval's per-query scoring work by 48.9× and SCT's by 6.4×.

Progressive descent itself is not supported. Retrieving the same residuals *without* the tree performs identically when the document is given ($p = 0.27$), and descent is substantially worse when the system must select the document (0.122 vs. 0.165, $p < 0.001$). Routing accuracy localises the cause: descent selects the correct paper 20.2% of the time against 39.3% for flat retrieval, because that choice is made from the root residual, the most compressed node in the tree. We conclude that the residual representation is worth keeping and top-down routing is not.

1 Introduction

Retrieval-Augmented Generation (RAG) has become the dominant paradigm for grounding large language models (LLMs) in external knowledge [Lewis et al., 2020, Guu et al., 2020]. The standard pipeline—chunk documents into fixed-size segments, embed them into a dense vector space [Karpukhin et al., 2020], and retrieve the top-k most similar chunks at query time—is conceptually simple and widely deployed [Gao et al., 2024]. However, this flat retrieval approach has three fundamental limitations that we address in this work.

Similarity ≠ relevance. Dense retrieval models measure semantic proximity in embedding space [Karpukhin et al., 2020, Izacard et al., 2022, Khattab and Zaharia, 2020], but proximity does not imply informational relevance. A passage about "temperature changes in the Arctic" is semantically *closer* to a query about "climate in Antarctica" than a passage about "policy responses to Arctic warming," yet the latter may be more *relevant* for answering "What policies address Arctic warming?" This distinction between similarity and relevance is well-documented in information retrieval [Thakur et al., 2021, Zhao et al., 2022].

Fixed granularity is suboptimal. Some queries require only a high-level summary ("What is this paper about?") while others demand specific details ("What F1 score did Model X achieve on dataset Y?"). Fixed-size chunking treats all queries identically. Recent work on adaptive chunking [Zhong et al., 2024, Günther et al., 2024] and hierarchical chunking [Zhang et al., 2025] has begun to address this, but these approaches still operate within a flat retrieval paradigm.

Document structure is discarded. Documents have inherent hierarchical organization—chapters, sections, subsections, paragraphs—that encodes semantic relationships and scoping. Traditional RAG pipelines destroy this structure during chunking [Wang et al., 2025]. The Hierarchical Attention Network [Yang et al., 2016] demonstrated the value of preserving document hierarchy for classification; we argue this principle extends to retrieval.

We propose **Semantic Compression Trees (SCT)**, a framework that addresses all three limitations. An SCT is a hierarchical tree where: (1) each node represents a unit of knowledge at a specific *resolution level*; (2) each node stores a *semantic residual*—only the information it contributes beyond its parent; and (3) retrieval is *progressive*—starting at the root and descending into relevant branches until accumulated context suffices. The key insight is that retrieval cost becomes proportional to query *specificity*, not corpus size.

Contributions. We make the following contributions:

1. We formalize the concept of *semantic residuals* for hierarchical knowledge representation, defining the SCT data structure with a guaranteed accumulation property (Section 3).
2. We present a source-agnostic tree construction algorithm with pluggable compression functions, supporting documents, databases, and plain text (Section 4).
3. We define progressive descent retrieval as a fixed-width beam over residual embeddings, with an optional similarity floor that halts descent when further detail is unlikely to help, and give its cost in nodes scored (Section 5).
4. We evaluate SCT on QASPER [Dasigi et al., 2021] under two protocols separating index quality from document routing, with generated-answer metrics, gold-evidence retrieval metrics, LLM-as-judge scoring [Zheng et al., 2023, Es et al., 2024], confidence intervals, and paired significance tests (Section 6).
5. We report a mixed result and separate its parts. Residual storage and the cost model hold up; progressive top-down descent does not, and we identify document routing from the root residual as the mechanism by which it fails (Section 7).
6. We show that the better compressor depends on the protocol — a zero-LLM extractive one when the document is given, an LLM one when it must be found — and that the mechanism behind the difference is visible in evidence recall rather than in answer scores alone (Section 7).

2 Related Work

Retrieval-Augmented Generation. RAG was introduced by Lewis et al. [2020], building on REALM [Guu et al., 2020], which first showed how to pre-train a knowledge retriever jointly with a language model. Fusion-in-Decoder [Izacard and Grave, 2021] demonstrated that generative models can effectively aggregate evidence from multiple retrieved passages. Subsequent work has explored when to retrieve: Self-RAG [Asai et al., 2024] trains models to adaptively decide whether retrieval is needed, while FLARE [Jiang et al., 2023] retrieves iteratively during generation when low-confidence tokens are produced. Recent surveys [Gao et al., 2024, Gupta et al., 2024] trace the evolution from "Naive RAG" (chunk-embed-retrieve) through "Advanced RAG" (query rewriting, re-ranking) to "Modular RAG" (composable pipelines). Our work introduces a fundamentally different indexing strategy orthogonal to these retrieval-time improvements.

Dense and Sparse Retrieval. Dense Passage Retrieval (DPR) [Karpukhin et al., 2020] established dual-encoder architectures for retrieval, outperforming BM25 by 9–19% on passage retrieval. ColBERT [Khattab and Zaharia, 2020] introduced late interaction for efficient yet effective retrieval, further improved by ColBERTv2 [Santhanam et al., 2022] with residual compression. Contriever [Izacard et al., 2022] showed that unsupervised contrastive pre-training can produce competitive retrievers without labeled data, while E5 [Wang et al., 2022] and GTR [Ni et al., 2022] demonstrated scaling benefits for embedding models. Zhao et al. [2022] provide a comprehensive survey covering 300+ papers on dense retrieval. These methods operate on fixed-size text segments; SCT is complementary—it structures the *index*, while any retriever can be used for child selection during progressive descent.

Hierarchical and Tree-Based Retrieval. RAPTOR [Sarthi et al., 2024] is the most closely related work. It recursively clusters and summarizes text chunks bottom-up, building a tree where each level stores a *full summary*. This enables multi-level retrieval but introduces significant redundancy: information captured in a parent summary is repeated in its children. SCT eliminates this redundancy through the semantic residual formulation. PageIndex [VectifyAI, 2025] uses LLM calls to construct table-of-contents-style trees from PDFs, but is limited to a single document format, requires 50–200+ LLM calls for construction with sequential verification loops, and does not formalize the relationship between tree levels. HiChunk [Zhang et al., 2025] employs fine-tuned LLMs for hierarchical document structuring with auto-merge retrieval. Chen et al. [2025] present coarse-to-fine retrieval using a concept hierarchy. The Hierarchical Re-ranker Retriever [Li et al., 2025] simultaneously exploits parent, intermediate, and sentence-level chunks. Unlike these approaches, SCT provides a formal residual framework with an accumulation guarantee and is source-agnostic.

Knowledge Graph Approaches. GraphRAG [Edge et al., 2024] extracts entity knowledge graphs with community summaries, achieving improvements on global sensemaking queries. HippoRAG [Gutierrez et al., 2024], inspired by hippocampal indexing theory, combines LLMs, knowledge graphs, and Personalized PageRank for multi-hop QA, reporting up to 20% improvement over existing RAG with 10–30× cost reduction. KG-RAG [Soman et al., 2024] integrates biomedical knowledge graphs with LLMs. These approaches capture entity-level relationships but do not provide the multi-resolution text access that SCT enables.

Chunking Strategies. The impact of chunking on RAG quality has received increasing attention. Mix-of-Granularity [Zhong et al., 2024] uses a router to dynamically determine optimal chunk size per query. Late Chunking [Günther et al., 2024] proposes embedding all tokens with full document context before segmenting, preserving long-range dependencies. Wang et al. [2025] systematically evaluate advanced chunking strategies, finding that traditional fixed-size chunking often fragments context and reduces coherence. SCT sidesteps the chunking problem entirely: boundaries follow the document's natural structure, and the residual formulation ensures no information is lost at segment boundaries.

Evaluation. LLM-as-judge evaluation has emerged as a scalable alternative to human assessment. Zheng et al. [2023] showed GPT-4 achieves >80% agreement with human preferences, while G-Eval [Liu et al., 2023] demonstrated chain-of-thought scoring for NLG. RAGAS [Es et al., 2024] provides reference-free RAG evaluation measuring faithfulness, answer relevance, and context relevance. Prometheus 2 [Kim et al., 2024] offers open-source judge models. We

Table 1: Comparison with related approaches.

System	Multi-res.	Non-redund.	Source-agn.	Formal	LLM-free
DPR/ColBERT	✗	N/A	Partial	✗	✓
RAPTOR	✓	✗	Docs only	✗	✗
PageIndex	Partial	✗	PDF only	✗	✗
GraphRAG	✗	N/A	Text only	✗	✗
HippoRAG	✗	N/A	Text only	Partial	✗
SCT (ours)	✓	✓[†]	✓	✓	✓[*]

[*]With extractive compression; LLM-based compression is also supported. [†]Non-redundancy is a property of the formulation, not a guarantee of the implementation: semantic subtraction is approximated, and we quantify the residual loss in Section 7.

adopt this paradigm for our evaluation, complementing it with standard token-level metrics [Rajpurkar et al., 2016].

Positioning. SCT differs from all prior work in three key ways: (1) the semantic residual formulation provides a principled, *non-redundant* multi-resolution representation with a formal accumulation guarantee; (2) the framework is source-agnostic, applying uniformly to documents, databases, and plain text; and (3) progressive retrieval makes query cost proportional to query specificity. Table 1 summarizes the comparison.

3 Semantic Compression Trees

3.1 Formal Definition

Let D be a knowledge source with text $T(D)$. A **Semantic Compression Tree** over D is a rooted tree $\mathcal{T} = (V, E)$ where each node $v \in V$ is associated with:

- $c(v)$: the original content of the source region represented by v
- $\ell(v) \in \{0, 1, \ldots, L\}$: the compression level (0 = root, most abstract)
- $\mathcal{R}(v)$: the *semantic residual* of v

Definition 1 (Semantic Residual)**.** *Let $\mathcal{C}(\cdot, \ell)$ be a compression function that produces a summary at level ℓ. The semantic residual $\mathcal{R}(v)$ is defined as:*

$$\mathcal{R}(v) = \begin{cases} \mathcal{C}(T(D), 0) & \textit{if } v = r \textit{ (root)} \\ c(v) \ominus \mathcal{C}(c(\mathrm{pa}(v)), \ell(\mathrm{pa}(v))) & \textit{otherwise} \end{cases} \tag{1}$$

where $\mathrm{pa}(v)$ *denotes the parent of v and $\ominus$ denotes* semantic subtraction*: extracting from $c(v)$ only information not captured in the compressed parent.*

Property 1 (Accumulation)**.** *For any node v with path from root $(r = v_0, v_1, \ldots, v_k = v)$:*

$$\mathrm{Context}(v) = \bigoplus_{i=0}^{k} \mathcal{R}(v_i) \tag{2}$$

where $\bigoplus$ denotes concatenation. The accumulated context reconstructs the semantic content at v's resolution level. At the root, context is maximally compressed; at leaves, it contains the full detail.

This formulation draws an analogy to wavelet transforms in signal processing: each level stores the "detail coefficients" (residuals) needed to reconstruct the signal at the next finer resolution. The root stores the "approximation coefficients"—the coarsest representation.

3.2 Compression Functions

We define the compression function $\mathcal{C}$ as a protocol with two operations:

1. $\mathcal{C}(t, \ell) \to s$: Compress text t to resolution level ℓ
2. $\text{Residual}(t_{\text{child}}, s_{\text{parent}}, \ell) \to r$: Extract semantic residual

We implement and evaluate two concrete strategies:

LLM-Based Compression. The LLM receives the parent summary and child content, and is prompted to extract only information in the child *not captured* by the parent. The system prompt instructs: "Extract ONLY the information in the child section that is NOT already captured in the parent summary. Preserve specific numbers, names, and dates." This produces the highest-quality residuals at the cost of one LLM call per node.

Extractive Compression. For each sentence in the child content, we compute a novelty score as the fraction of unique terms (after lowercasing and punctuation removal) not present in the parent summary. Sentences above an adaptive threshold are retained. This requires *zero LLM calls* and runs in linear time.

4 Tree Construction

4.1 Source-Agnostic Parsing

SCT construction begins with parsing the source into an intermediate `HierarchyNode` tree. This decouples source-specific structure detection from the universal residual computation:

- **Markdown/Documents:** Heading levels (H1–H6) define the hierarchy.
- **PDFs:** Font-size analysis identifies headings; the dominant non-body font size defines section boundaries.
- **SQL Databases:** Schema overview → table details → column statistics → sample rows.
- **Plain Text:** Paragraph segmentation with Jaccard similarity boundary detection for topic shifts.

Adding a new source type requires only implementing the parser interface; the construction and retrieval algorithms remain unchanged.

4.2 Construction Algorithm

Complexity. Construction requires $O(N)$ compression calls for N nodes. Crucially, all siblings at the same level are independent—they share the same parent summary but their residual computations do not depend on each other. This enables parallelization across siblings, reducing wall-clock time by a factor of the branching factor. In contrast, PageIndex [VectifyAI, 2025] requires sequential verification-and-fix loops with $O(\text{pages} \times \text{retries})$ LLM calls.

5 Progressive Retrieval

Given an SCT $\mathcal{T}$ and a query q, retrieval proceeds by progressive descent (Algorithm 2).

Descent is a fixed-width beam. At each level the candidate set is the union of the current frontier's children; every candidate is scored by cosine similarity between the query embedding and its residual embedding, and the k highest are retained. Taking the top-k over the union rather than per parent is what bounds the frontier: selecting k children of every frontier node would multiply the frontier by k at each level, making cost exponential in depth.

The optional floor τ implements resolution-aware retrieval. When no candidate at a level is sufficiently similar to the query, further detail is unlikely to help and descent halts, so a broad query is answered from the upper levels at proportionately lower cost. The floor is not applied

Algorithm 1 BuildSCTree: Construct a Semantic Compression Tree

Require: Parsed hierarchy H, compression function $\mathcal{C}$, max depth D
$s_r \leftarrow \mathcal{C}(\text{FullText}(H), 0)$ {Root compression}
Create root r; set $\mathcal{R}(r) \leftarrow s_r$
BuildSubtrees(r, s_r, H.children, 1, D)
return $\mathcal{T}$

Require: Parent p, parent summary s_p, children $\{h_1, \ldots, h_m\}$, depth d, max D
if $d > D$ **then**
  return
end if
for $i = 1$ **to** m **in parallel (siblings are independent) do**
  $r_i \leftarrow \text{Residual}(c(h_i), s_p, d)$ {Semantic subtraction}
  $s_i \leftarrow \mathcal{C}(c(h_i), d, s_p)$ {Summary for children}
  Add node h_i to $\mathcal{T}$ as child of p
  BuildSubtrees(h_i, s_i, h_i.children, $d + 1$, D)
end for

Algorithm 2 ProgressiveRetrieval

Require: Tree $\mathcal{T}$, query q, max depth D, beam width k, floor τ
$\text{ctx} \leftarrow \mathcal{R}(\text{root})$; $F \leftarrow \{\text{root}\}$
for $d = 1$ **to** D **do**
  $C \leftarrow \bigcup_{v \in F} \text{Children}(v)$
  if $C = \emptyset$ **then**
    break
  end if
  $s(v) \leftarrow \cos\big(\phi(q), \phi(\mathcal{R}(v))\big)$ for all $v \in C$
  if $d > 1$ **and** $\max_{v \in C} s(v) < \tau$ **then**
    break
  end if
  $F \leftarrow \text{top-}k\big(C, s\big)$ {over the union, not per parent}
  $\text{ctx} \leftarrow \text{ctx} \oplus \bigoplus_{v \in F} \mathcal{R}(v)$
end for
return ctx

at the first level, since every query must be answered from somewhere. We report τ disabled as the default configuration and enabled as a separate variant.

We additionally evaluate two deterministic, query-independent modes as references: **fixed-depth**, which takes all children to a given depth, and **exhaustive**, which takes all children to the depth bound and therefore returns the whole subtree. Neither consults the query, so neither is a retrieval strategy; they bound what the index makes available.

Descent scores at most k children per frontier node per level, so it visits $O(D \cdot k \cdot b)$ nodes and accumulates at most $D \cdot k$ residuals, where b is the branching factor and D the depth bound—in both cases independent of the number of documents indexed. Flat retrieval scores all N units. Under the corpus protocol of Section 6.2, where document selection is the first level of descent, the root scan adds a term linear in the number of documents, giving $O(N_{\text{docs}} + D \cdot k \cdot b)$; that term is a nearest-neighbour lookup over document roots and is amenable to standard approximate indexing, which we do not pursue here.

6 Experiments

6.1 Dataset

QASPER [Dasigi et al., 2021] contains 1,585 NLP papers with 5,049 information-seeking questions written by practitioners from abstracts alone and answered from the full text. Two properties make it suitable here. Papers carry a real section hierarchy, which is what a structure-preserving index consumes; and each answer is annotated with the source paragraphs supporting it, which allows retrieval to be scored directly rather than only through a generated answer.

We evaluate on the first 50 papers of the test split, giving 173 answerable questions (101 extractive, 46 free-form, 26 yes/no); 6 questions annotated unanswerable are excluded. Gold evidence is available for 170 of the 173.

QASPER encodes section nesting inside the section name, using a `" ::: "` separator: `"Datasets ::: AntiScam Dataset"` is a subsection of `"Datasets"`. We reconstruct the heading levels from this convention. Rendering each entry at a single heading level instead collapses every paper to a two-level tree, which would make a depth-sensitivity experiment vacuous. With nesting preserved, 22 of the 50 papers contain genuine subsections and trees reach depth 4.

Each paper is one document. We deliberately do not emit section-level pseudo-documents alongside the papers: duplicating the same text at two granularities inside a flat baseline's index inflates its candidate pool with near-identical units and makes the comparison a comparison of segmentation choices. Every system receives the same document set and performs its own segmentation.

6.2 Evaluation protocols

Retrieval systems are given a query and, in one of the two protocols, the identifier of the document the question concerns. They receive nothing else. No reference answer and no evidence annotation is visible to any system at any point; gold data is read only after retrieval has returned, by the scoring code.

P1: single-document (primary). The benchmark supplies the paper for each question and the system retrieves within it. This is the standard QASPER setting, so results are comparable to published numbers, and it isolates index quality from document routing. Each system builds an index scoped to the single paper rather than filtering a corpus-wide index after the fact, which would leak cross-document statistics—corpus-wide inverse document frequency, clusters spanning papers—into a single-document result.

P2: corpus (secondary). All 50 papers are pooled and the document identifier is withheld. A system must locate the relevant paper and the relevant passage within it from the query alone. This is the setting in which the claim that cost tracks query specificity rather than corpus size is meaningful. For SCT, document selection requires no additional machinery: the per-paper trees are joined under a virtual root, so the corpus becomes one further level of the same hierarchy and Algorithm 2 performs routing and descent in one pass.

6.3 Systems

All systems that segment text use 512-token windows with 64-token overlap, so sparse and dense baselines differ only in their scoring function. All flat retrievers return their top 5 units.

- **BM25**: Okapi BM25 [Robertson and Zaragoza, 2009], $k_1 = 1.2$, $b = 0.75$, over chunks.
- **VectorRAG**: dense retrieval over the same chunks, embedded with `text-embedding-3-small` and ranked by cosine similarity. The dominant production pattern [Karpukhin et al., 2020, Izacard et al., 2022], and our reference system for all significance tests.

- **VectorRAG (top-1)**: the same system returning a single chunk. A token-budget comparator, included because SCT returns less text than top-5 dense retrieval and a quality difference at unequal context size is partly a difference in context size.
- **RAPTOR** [Sarthi et al., 2024]: bottom-up clustering and LLM summarisation, repeated until one node remains; retrieval scores all nodes, leaves and summaries alike, by cosine similarity. The closest prior work.
- **SCT-Extractive**: SCT with the zero-LLM extractive compressor, embedding-guided descent, beam width $k = 3$, depth bound $D = 5$.
- **SCT-LLM**: SCT with LLM compression, otherwise identical.

Ablations isolate the two claims of the method. **SCT-FullSummary** stores a full compressed summary at each node instead of a residual, holding the tree fixed, and so isolates the residual formulation. **SCT-Flat** computes residuals and then discards the tree, retrieving the residual texts by flat cosine similarity, and so isolates the hierarchy. **SCT-EarlyStop** enables the descent floor at $\tau = 0.30$. **SCT-Depth-k** varies the depth bound over $\{1, 2, 3, 5\}$.

6.4 Metrics

Answer quality. Answer F1 and Exact Match on the *generated* answer, using SQuAD normalisation [Rajpurkar et al., 2016] and scored against the best-matching annotator reference, as QASPER prescribes for multiply-annotated questions. Answers are generated by prompting GPT-5.4 with the retrieved context and the question, instructed to answer from the context alone.

Retrieval quality. Answer metrics see retrieval only through a generator, so we also score the context directly. **Evidence recall** is the fraction of gold evidence paragraphs present verbatim in the retrieved context. **Evidence token recall** is the fraction of gold evidence *tokens* present. Reporting both separates two failures that verbatim recall alone conflates: routing to the wrong part of the document, and routing correctly but compressing the evidence away—the second being an inherent cost of any summarising index, and one SCT incurs by construction. **Evidence unit F1** scores precision over retrieved units, so padding the context is penalised.

Cost. Context tokens supplied to the generator, measured with the `o200k_base` tokeniser; index nodes scored per query; retrieval latency; and index construction cost in LLM calls. We report context tokens alongside every quality number, because retrieval quality at unequal context size is not a single comparison.

LLM-as-judge. Following Zheng et al. [2023] and Es et al. [2024], GPT-5.4 scores context relevance, context completeness, answer correctness, and answer faithfulness on a 1–5 rubric normalised to $[0, 1]$, at temperature 0, one independent call per dimension following Liu et al. [2023]. Judge scoring is run on the full question set, not a subset.

Statistics. Every headline metric carries a bootstrap 95% confidence interval, and every comparison against the reference system carries a paired bootstrap p-value, both over 10,000 resamples with a fixed seed.

Excluded questions. The provider's content filter refuses to generate an answer for a small number of (question, context) pairs, since QASPER indexes NLP papers and those include research on hate speech and offensive language. The filter responds to the retrieved context, so it triggers on different questions for different systems; scoring a refusal as an incorrect answer would penalise whichever system retrieved the relevant passage. We therefore report answer

Table 2: Main results, single-document protocol ($n = 172$ questions, 50 QASPER papers). Answer F1 is token F1 of the *generated* answer against the best-matching annotator reference. Brackets give bootstrap 95% confidence intervals; p is a paired bootstrap test against VectorRAG. Context tokens are the generator's input, reported so quality is read against cost; nodes are index nodes scored per query.

System	Ans. F1	95% CI	p	Ev. Rec.	Ev. Tok. Rec.	Ctx. tok.
BM25	0.262	[0.233, 0.290]	0.024	0.794	0.946	3,165
VectorRAG (top-5)	**0.277**	[0.248, 0.306]	—	0.866	**0.969**	3,120
VectorRAG (top-1)	0.232	[0.203, 0.262]	<0.001	0.470	0.759	646
RAPTOR	0.272	[0.243, 0.301]	0.207	0.794	0.955	3,058
SCT-Extractive (ours)	0.274	[0.244, 0.306]	0.366	0.720	0.895	2,178
SCT-LLM (ours)	0.233	[0.205, 0.261]	<0.001	0.009	0.590	802

Table 3: Main results, corpus protocol ($n = 173$ questions, 50 QASPER papers). Answer F1 is token F1 of the *generated* answer against the best-matching annotator reference. Brackets give bootstrap 95% confidence intervals; p is a paired bootstrap test against VectorRAG. Context tokens are the generator's input, reported so quality is read against cost; nodes are index nodes scored per query.

System	Ans. F1	95% CI	p	Ev. Rec.	Ev. Tok. Rec.	Ctx. tok.
BM25	0.154	[0.129, 0.179]	0.138	0.339	0.750	3,214
VectorRAG (top-5)	**0.165**	[0.139, 0.192]	—	0.390	**0.767**	3,206
VectorRAG (top-1)	0.143	[0.118, 0.168]	<0.001	0.205	0.516	631
RAPTOR	0.156	[0.131, 0.181]	0.048	0.365	0.760	3,425
SCT-Extractive (ours)	0.122	[0.100, 0.145]	<0.001	0.173	0.575	2,187
SCT-LLM (ours)	0.137	[0.114, 0.162]	0.010	0.006	0.472	1,042

metrics over the questions scoreable for every system, and state the number excluded with each table. Retrieval metrics are unaffected and use all questions.

6.5 Results

The document is given (P1). Table 2 reports the primary protocol. SCT-Extractive reaches 0.274 Answer F1 against dense retrieval's 0.277, with a paired bootstrap $p = 0.366$ and confidence intervals that almost coincide ($[0.244, 0.306]$ and $[0.248, 0.306]$). This is a tie, and we report it as one. What differs is cost: SCT supplies 2,178 context tokens to the generator against 3,120, a 30% reduction, and its index requires no LLM calls to construct. RAPTOR, the closest prior work, also ties the reference (0.272, $p = 0.207$) but needs 131 LLM calls to build its summaries. BM25 is the only baseline the reference separates from (0.262, $p = 0.024$).

Reducing dense retrieval's context budget to match SCT's does not preserve its quality: VectorRAG at top-1 supplies 646 tokens and drops to 0.232 ($p < 0.001$), well below SCT at 2,178 tokens. Fewer tokens is not by itself an advantage; the question is what is in them.

The document must be found (P2). Table 3 withholds the document identifier. Every ordering from P1 survives except SCT's: SCT-Extractive falls to 0.122 against the reference's 0.165 ($p < 0.001$), the largest gap in either table. Routing accuracy identifies the cause directly. SCT descent selects the correct paper for 20.2% of questions; BM25, dense retrieval, and RAPTOR all select it for 38.7–39.3%.

Judge scores. Table 4 shows the judge ranking answer correctness as dense retrieval (0.780), RAPTOR (0.743), SCT-Flat (0.737), BM25 (0.711), SCT-Extractive (0.688). Token-level F1

Table 4: LLM-as-judge scores, single-document protocol ($n = 172$), rubric-scored 1–5 and normalised to $[0, 1]$. Judge dimensions are scored independently, one call each.

System	Relev.	Compl.	Correct.	Faith.
BM25	0.760	0.903	0.711	0.994
VectorRAG (top-5)	0.794	0.923	**0.780**	0.996
VectorRAG (top-1)	0.664	0.690	0.597	0.996
RAPTOR	**0.805**	**0.933**	0.743	0.974
SCT-Extractive (ours)	0.785	0.847	0.688	**0.997**
SCT-LLM (ours)	0.759	0.817	0.674	0.994
SCT-FullSummary	0.626	0.557	0.432	0.996
SCT-Flat	0.795	0.879	0.737	0.993

Table 5: Ablations and depth sweep, single-document protocol. Δ is the change in Answer F1 against the full system. Nodes visited counts index nodes scored per query.

Variant	Ans. F1	95% CI	Δ	Ev. Tok.	Ctx. tok.	Nodes
Component ablations						
SCT-Extractive (ours)	0.274	[0.244, 0.306]	—	0.895	2,178	13
SCT-FullSummary	0.205	[0.179, 0.233]	-0.069	0.584	437	12
SCT-Flat	0.272	[0.244, 0.302]	-0.002	0.939	2,149	17
SCT-EarlyStop	0.273	[0.243, 0.304]	-0.002	0.895	2,098	13
Retrieval depth bound						
Depth = 1	0.272	[0.242, 0.303]	-0.003	0.895	1,744	10
Depth = 2	0.273	[0.242, 0.304]	-0.002	0.895	2,124	12
Depth = 3	0.274	[0.244, 0.306]	+0.000	0.895	2,177	13
Depth = 5	0.274	[0.244, 0.306]	+0.000	0.895	2,178	13

places SCT level with the reference while the judge places it below. We report both rather than selecting the more favourable: the two metrics measure different things, and their disagreement is part of the result. Faithfulness is uniformly high (0.974–0.997) and does not discriminate between systems on this benchmark.

Ablations. Table 5 isolates the components and Table 6 tests each variant directly against the configuration it modifies. The direct tests matter here: Table 2 compares every system to one reference, which cannot establish a claim about two variants relative to *each other*, since two systems both indistinguishable from a third may still differ. Two results matter and they point in opposite directions.

Storing residuals rather than full summaries at each node is worth 0.069 F1 (0.274 vs. 0.205, $p < 0.001$), a 34% relative gain — the clearest support for any component of the method. Part of that gap is a difference in context volume: full summaries at each level retain less text (437 tokens against 2,178), so this establishes that residuals preserve more of what matters, not that they are better at an equal token budget.

Discarding the tree costs nothing. SCT-Flat holds the residual content fixed and retrieves it by flat cosine similarity; the paired difference is +0.0017 in the tree's favour with $p = 0.420$ and a 95% interval of $[-0.014, +0.017]$. The interval is narrow and centred on zero, so this is an absence of effect rather than an inconclusive test: the hierarchy is not contributing to answer quality when the document is given.

Descent depth. The depth sweep moves Answer F1 from 0.2715 at depth 1 to 0.2741 at depth 5, while context grows from 1,744 to 2,178 tokens. Depth 3 and depth 5 are identical because

Table 6: Direct paired comparisons between SCT variants on Answer F1. Δ is A minus B, so a positive value favours A; p is a paired bootstrap test and the interval is a bootstrap 95% CI on Δ itself. Tested pairwise because the claims are pairwise: two systems that are each indistinguishable from a common reference may still differ from one another.

Comparison (A vs. B)	Δ	p	**95% CI of** Δ
Single-document protocol			
Tree vs. no tree (identical residuals)	+0.0017	0.420	[-0.0138, +0.0168]
Residuals vs. full summaries	+0.0688	<0.001	[+0.0411, +0.0983]
Extractive vs. LLM compression	+0.0411	<0.001	[+0.0207, +0.0618]
Depth = 5 vs. depth = 1	+0.0026	0.161	[-0.0025, +0.0075]
Corpus protocol			
Tree vs. no tree (identical residuals)	-0.0255	0.009	[-0.0475, -0.0042]
Residuals vs. full summaries	+0.0221	0.003	[+0.0056, +0.0403]
Extractive vs. LLM compression	-0.0154	0.040	[-0.0335, +0.0016]
Depth = 5 vs. depth = 1	+0.0006	0.332	[-0.0022, +0.0034]

Table 7: Index construction cost over 50 papers, and per-query retrieval time. LLM calls are the model-independent construction cost. Build times are wall clock at the stated concurrency. Retrieval time is reported for scale only and must not be compared across systems: SCT scores residual embeddings in pure Python while the flat baselines use a vectorised dot product, so the difference measures the implementations. Table 8 gives the implementation-independent comparison.

Protocol	**System**	**LLM calls**	**Units indexed**	**Build (s)**	**Retr. (ms)**
Single-doc	BM25	0	386	0.1	0.0
	VectorRAG (top-5)	0	386	0.1	0.4
	VectorRAG (top-1)	0	386	0.1	0.2
	RAPTOR	131	517	0.2	0.2
	SCT-Extractive (ours)	0	814	0.3	1.3
	SCT-LLM (ours)	0	814	3.1	1.4
Corpus	BM25	0	386	0.0	0.3
	VectorRAG (top-5)	0	386	0.1	0.4
	VectorRAG (top-1)	0	386	0.1	0.2
	RAPTOR	99	485	8.2	0.2
	SCT-Extractive (ours)	0	814	0.4	7.3
	SCT-LLM (ours)	0	814	0.4	7.3

few trees are deeper than three levels: although subsection nesting is preserved, 28 of the 50 papers contain no subsections at all. One level of descent captures nearly all of the available benefit on this corpus. Enabling the similarity floor ($\tau = 0.30$) reduces context to 2,098 tokens with Answer F1 unchanged at 0.273, so early stopping is close to free but not, at this threshold, a large saving.

Cost. Table 8 gives the result that holds most cleanly. Growing the indexed collection from one document to fifty multiplies the number of index nodes scored per query by 48.9× for flat retrieval and by 6.4× for SCT: 386 nodes against 81. Retrieval cost is much closer to independent of collection size, as the formulation predicts.

We report nodes scored rather than latency for this comparison. Table 3 lists SCT at 7.3 ms against dense retrieval's 0.4 ms, but SCT scores residual embeddings in pure Python while the baselines use a vectorised dot product; that comparison measures implementations. Index construction cost is in Table 7.

Table 8: Index nodes scored per query, as the indexed collection grows from one document to fifty. Reported instead of latency because SCT scores residual embeddings in pure Python while the flat baselines use a vectorised dot product; a wall-clock comparison would measure the implementations. Growth is the ratio between the two columns.

System	**1 document**	**50 documents**	**Growth**
BM25	8	386	48.9×
VectorRAG (top-5)	8	386	48.9×
VectorRAG (top-1)	8	386	48.9×
RAPTOR	11	485	46.0×
SCT-Extractive (ours)	13	81	6.4×
SCT-LLM (ours)	13	80	6.2×

Table 9: Context–answer overlap F1, a diagnostic rather than a quality measure, shown against context length. Overlap F1 divides by the length of the retrieved context, so it rewards returning less text whether or not that text answers the question. Answer F1 is repeated for contrast.

System	**Ctx.–Ans. overlap F1**	**Ctx. tokens**	**Answer F1**
BM25	0.010	3,165	0.262
VectorRAG (top-5)	0.011	3,120	0.277
VectorRAG (top-1)	0.038	646	0.232
RAPTOR	0.011	3,058	0.272
SCT-Extractive (ours)	0.023	2,178	0.274
SCT-LLM (ours)	0.034	802	0.233

The metric matters more than the method. Table 9 reports token overlap between the retrieved context and the gold answer, a metric sometimes used as a cheap proxy for retrieval quality. Its precision term divides by context length, so it rises as context shrinks regardless of whether the context answers anything: dense retrieval at top-1 scores 3.4× dense retrieval at top-5 on this measure while scoring *lower* on Answer F1. A system evaluated on context-overlap F1 can therefore be improved by returning less text. We include the table because the failure mode is not obvious from the definition, and because any retrieval comparison drawn from it will be dominated by context length.

7 Analysis

Why routing fails, and why it is structural. Under progressive descent, choosing a document is the first selection the algorithm makes, and it is made by comparing the query against each document's root residual. The root residual is by construction the most compressed node in the tree: one to two sentences standing in for an entire paper. A query asking which optimiser was used in a specific ablation has almost nothing to match against in such a summary.

Flat retrieval cannot make this error, because it never summarises before comparing — the query is scored against every passage, and a passage mentioning the optimiser is reachable directly. This is not a tuning deficiency in our implementation. It is a consequence of ordering compression before selection, and it applies to any strictly top-down index over a collection.

The controlled comparison is SCT-Flat, which fixes the residual content and changes only the retrieval strategy: routing accuracy rises from 0.202 to 0.347 and Answer F1 from 0.122 to 0.147, a paired difference of -0.0255 against the tree ($p = 0.009$, 95% CI $[-0.047, -0.004]$). The hierarchy, not the residual representation, is what costs the accuracy.

It is worth being precise about what this does and does not reject. The tree is indispensable at *construction* time — a residual is defined against a parent, so there is no residual without the hierarchy that produces it, and the residual is the component that wins. What these experiments reject is the narrower proposition that descending that hierarchy is a good way to *retrieve* from it.

Why the depth sweep is flat, and why that is informative. All depth settings score within 0.0006 F1 of one another under P2, and routing accuracy is 0.202 for every one of them. Once the first level has selected the wrong document, no amount of further descent can recover: depth is conditionally irrelevant given a routing error, and routing errs on four questions in five. Under P1, where routing cannot fail, depth still buys only +0.0026 F1 for a 25% increase in context. The multi-resolution argument requires documents deep enough to have multiple resolutions, and scientific papers largely do not.

Compression strategy: the free option wins. SCT-LLM underperforms SCT-Extractive (0.233 vs. 0.274) while costing substantially more to build. The evidence metrics locate the failure precisely. Verbatim evidence recall collapses to 0.009 — essentially no gold paragraph survives an LLM-compressed residual intact — while token-level recall holds at 0.590. The compressor is finding roughly the right region and then paraphrasing away the text that constitutes the evidence. For extractive compression the same pair is 0.720 and 0.895: it also loses source text, but far less of it.

This is a general point about summarising indexes rather than a fact about one compressor. Answer-level metrics alone would have shown a modest deficit and invited a prompt-tuning explanation; the recall pair shows that the mechanism is lossy paraphrase, which prompt tuning does not fix.

The ordering reverses under the corpus protocol, where SCT-LLM beats SCT-Extractive by 0.0154 ($p = 0.040$). Routing explains the reversal: an LLM-written root summary is a better description of a paper than two extracted sentences, so it routes more accurately (0.358 against 0.202) — and once document selection is the binding constraint, that advantage outweighs the evidence the compressor destroys. The two protocols therefore recommend different compressors, which is itself an argument for reporting both rather than one.

What the residual formulation is buying. Against full summaries at each node, residuals gain 34% relative F1. The mechanism is visible in the evidence metrics: full summaries reach only 0.152 verbatim and 0.584 token recall against 0.720 and 0.895 for residuals. Storing what a node *adds* preserves specifics — numbers, names, dataset sizes — that summarising the node in isolation discards, because those specifics are exactly what a summary judges unimportant relative to the section's main point. This is the component of the method we would keep.

Where this leaves the design. The two findings compose into a concrete recommendation. Residual storage is worth keeping: it matches dense retrieval at 70% of the context budget with no LLM indexing cost. Strictly top-down routing is not: it halves document selection accuracy for a cost saving the accuracy loss exceeds on this benchmark. The design these results point to is a hybrid — select passages flat, then use the tree for resolution control within the selected document, keeping the residual representation and discarding top-down routing. We have not evaluated it and make no claim for it here.

Absolute numbers. Answer F1 near 0.27 under P1 is in the range reported for QASPER, which is a hard benchmark: questions are written from abstracts by readers who have not seen the full text, and many require synthesis across sections. Exact match is 0.000 for every system, which is expected rather than anomalous — a generated sentence almost never matches a short reference span byte for byte — and we therefore do not report it as a discriminating metric.

8 Discussion

Compression before selection. The clearest lesson from these experiments is about ordering. Any index that summarises before it selects must make its earliest and most consequential decision from its least informative representation. Progressive descent takes this to its limit: the choice of document is made from a one-to-two-sentence root residual, and a wrong choice cannot be recovered at any later level. Flat retrieval inverts the order — select first, over full passages — and pays for it in per-query work that grows with the collection.

Framed this way, the trade-off is not specific to SCT. It applies to RAPTOR's summary levels, to table-of-contents indexes, and to any router that operates on document-level abstractions. Our contribution here is a measurement of the price: on QASPER, moving document selection onto compressed representations halves routing accuracy, and the resulting quality loss exceeds the 7.6× reduction in nodes scored that it buys.

Relation to multi-resolution analysis. The residual formulation bears a structural analogy to wavelet decomposition, where each level stores detail coefficients for a frequency band and the reconstruction formula plays the role of our accumulation property (Property 1). The analogy is suggestive rather than formal, and the disanalogy is instructive: wavelet decomposition is invertible, whereas semantic subtraction is lossy, and our evidence-recall measurements quantify by how much. An information-theoretic account of semantic compression — in which that loss is bounded rather than merely measured — remains open.

Beyond documents. The construction is source-agnostic: our SQL parser maps a database schema to the same structure (database → tables → columns → row samples), and unlike prose this hierarchy is exact rather than inferred. Structured sources are also the case where our negative result may not transfer. A schema hierarchy has genuine depth and precise node boundaries, and a table name is a far better routing key than a paper's one-sentence summary. Whether descent is competitive there is an empirical question we have not answered, and it is the setting we would test next.

Future directions. Three follow-ups are implied directly by these results. **Hybrid retrieval**: select passages flat, then descend within the selected document, keeping the residual representation and discarding top-down routing. **Better routing keys**: score a document by its level-one residuals rather than its root, or index roots with an approximate nearest-neighbour structure, which would also remove the linear term from the cost bound. **Deeper corpora**: evaluate where documents have real depth — books, standards, legal codes, database schemas — since more than half our papers have no subsections for the depth bound to act on.

9 Conclusion

We presented Semantic Compression Trees, a hierarchical index in which each node stores only its semantic residual, together with a progressive descent retrieval algorithm and an evaluation on QASPER under two protocols that separate index quality from document routing.

The representation holds up and the retrieval strategy does not. When the relevant document is supplied, SCT with a zero-LLM extractive compressor matches dense retrieval on answer quality (0.274 vs. 0.277, $p = 0.37$) using 30% fewer context tokens and no LLM calls to build the index, and residual storage beats storing full summaries at each node by 34% relative ($p < 0.001$). Increasing the collection fifty-fold multiplies flat retrieval's per-query scoring work by 48.9× and SCT's by 6.4×, confirming that descent decouples query cost from collection size.

Progressive descent, however, contributes nothing over flat retrieval of the same residuals when the document is given ($p = 0.27$), and loses substantially when the document must be found (0.122 vs. 0.165, $p < 0.001$). The mechanism is document routing: descent selects the correct paper 20.2% of the time against 39.3% for flat retrieval, because that selection is made from the most compressed node in the tree. Depth beyond one level is worth +0.003 F1 for 25% more context. The choice of compressor depends on which protocol applies: the free extractive variant is the better one when the document is given, since LLM compression collapses verbatim evidence recall to 0.009, but LLM compression wins under the corpus protocol because its root summaries route more accurately.

The result we would build on is the residual representation, not the descent.

Limitations

Scale and single dataset. We evaluate on 50 QASPER papers and 173 questions. That is enough to place confidence intervals on the differences we report, and the intervals are wide enough that several comparisons are not resolved at this sample size; we say so where that is the case rather than reading a point estimate as a result. One dataset also cannot separate properties of SCT from properties of NLP papers, whose section conventions are unusually regular. NarrativeQA [Kocisky et al., 2018], HotpotQA [Yang et al., 2018], and Natural Questions [Kwiatkowski et al., 2019] would each stress a different assumption, and multi-hop questions in particular would test whether a single descent path is the right shape for evidence that must be assembled from several branches.

Structure dependence. The tree is derived from the document's own headings, so SCT inherits whatever structure the source provides. QASPER papers are close to a best case: explicit, consistently nested sections. Even here, 28 of 50 papers have no subsections at all, so the depth bound has nothing to act on for more than half the corpus — a ceiling on how much a multi-resolution argument can be demonstrated on this dataset. Unstructured sources (scanned PDFs, transcripts, chat logs) would fall back on inferred boundaries, which we do not evaluate.

Residual quality and its cost. Semantic subtraction ($\ominus$) is approximated, not exact. Extractive compression retains sentences that partially restate the parent; LLM compression can discard novel but low-salience detail. The consequence is measurable rather than hypothetical: the gap between verbatim and token-level evidence recall quantifies how much source text a residual fails to preserve, and it is the clearest cost of the representation. Applications that need exact source wording should read that gap, not the faithfulness score. A formal measure of residual quality remains open.

Top-down routing is bounded by the root summary. Under the corpus protocol, document selection is the first level of descent and therefore depends on the root residual, which is by construction the most aggressively compressed node in the tree — one to two sentences standing in for an entire paper. A flat retriever compares the query against every passage and so cannot make this class of error at all. Whether a top-down index can route as well as an exhaustive scan is a question about the first level, not about the method's descent, and a hybrid that routes flat and descends hierarchically is the obvious thing to try next. We do not attempt it here.

Judge-based metrics. The judge is a single model scoring with a rubric, and it is the same model family used to generate the answers, which risks self-preference. We report judge scores beside token-level and evidence-level metrics for exactly this reason, and note where they disagree; we do not treat agreement between them as validation, nor disagreement as noise.

Latency measurement. Reported retrieval latency excludes the query-embedding round trip, which is served from a cache uniformly for every embedding-based system. This isolates retrieval computation — the quantity the cost analysis concerns — from a network constant that is identical across those systems, but it means the figures are not end-to-end serving latency.

Reproducibility

Each experiment writes a report recording the code revision that produced it, the deployment names, the tokeniser, the bootstrap seed, per-question records for every system, and the measured API usage of the run. Every table in this paper is generated from those reports by script rather than transcribed, so a table cannot diverge from the run behind it.

Answer generation and judge scoring depend on a hosted model and are therefore not bitwise reproducible from scratch. Every API response is cached under a content hash of its request, and we re-ran both protocols against that cache: the confirmation pass reissued 58 of roughly 14,900 calls and reproduced every reported figure exactly. Retrieval, tree construction, extractive compression, and all statistics are deterministic given a fixed seed.

Two facts about the harness are worth stating explicitly, because they are the kind of detail that decides whether a retrieval result means anything. **First**, systems are driven through an interface that exposes only the query and, under P1, the document identifier; a reference answer or evidence annotation cannot reach a retrieval code path, and the harness carries regression tests that assert this, including one that reconstructs an answer-maximising selector and confirms the tests reject it. **Second**, retrieval is executed sequentially while generation and judging are parallelised, because retrieval latency is a reported quantity and would otherwise measure thread contention.

Ethics Statement

Environmental Impact. SCT-LLM construction issues LLM calls proportional to the number of tree nodes; the per-run figure is recorded in Table 7. The extractive variant was designed and evaluated as a zero-LLM-cost alternative precisely so that indexing cost is a choice rather than a fixed price, and we report what that choice costs in retrieval quality rather than assuming it is free.

Faithfulness and Misinformation. Any summarising index can distort what it compresses, and SCT compresses by construction: a residual is lossy with respect to its source. We measure this from two directions—judge-scored faithfulness of generated answers, and the gap between verbatim and token-level evidence recall, which quantifies how much source text a residual fails to preserve. For applications where exact source wording carries legal or clinical weight, that gap, not the faithfulness score, is the number to read.

Content filtering. A small number of questions could not be scored because the answer-generation provider refused the prompt on content-policy grounds; QASPER indexes NLP papers, including research on hate speech and offensive language. We report which questions were excluded and apply the exclusion identically to every system, since the filter responds to retrieved content and would otherwise penalise the systems that retrieved the relevant passage.

Data. All experiments use the publicly available QASPER dataset [Dasigi et al., 2021], distributed under CC BY 4.0. No private or personally identifiable data was used.

Acknowledgments

We thank the QASPER [Dasigi et al., 2021] authors for making their dataset publicly available and the Allen Institute for AI for hosting it.

About the Author

Junaid Farooq works at the boundary between AI research and production systems, with over a decade in artificial intelligence, machine learning, and software engineering.

He holds a PhD in Artificial Intelligence (2023) from the Artificial Intelligence Laboratory, Department of Electrical Engineering, National Institute of Technology Srinagar, where his doctoral work developed hybrid deep-learning architectures for spatiotemporal forecasting. He has written several peer-reviewed papers and two books: *The Art of Code: Tactics and Principles of Clean Code and Architecture* (2024), on engineering rigour and maintainable architecture, and *Zero Day: The Invisible War — Mastering Vulnerability in the Age of Artificial Intelligence* (2025), on security and safety for AI systems.

He is Chief Architect and Vice President of Artificial Intelligence at Sprouts.ai, where he leads platform architecture and production LLM and agentic systems.

References

Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of ICLR*, 2024.

Beeferman, D., Berger, A., and Lafferty, J. Statistical models for text segmentation. *Machine Learning*, 34:177–210, 1999.

Chalkidis, I., Dai, X., Fergadiotis, M., Malakasiotis, P., and Elliott, D. An exploration of hierarchical attention transformers for efficient long document classification. *arXiv preprint arXiv:2210.05529*, 2022.

Chen, X., et al. Hierarchical semantic retrieval with Cobweb. *arXiv preprint arXiv:2510.02539*, 2025.

Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., and Gardner, M. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of NAACL*, 2021.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., and Larson, J. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Es, S., James, J., Espinosa-Anke, L., and Schockaert, S. RAGAS: Automated evaluation of retrieval augmented generation. In *Proceedings of EACL (Demo Track)*, 2024.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., and Wang, H. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2024.

Günther, M., Mohr, I., Williams, D. J., Wang, B., and Xiao, H. Late chunking: Contextual chunk embeddings using long-context embedding models. *arXiv preprint arXiv:2409.04701*, 2024.

Gupta, S., Ranjan, R., and Singh, S. N. A comprehensive survey of retrieval-augmented generation (RAG): Evolution, current landscape and future directions. *arXiv preprint arXiv:2410.12837*, 2024.

Gutierrez, B. J., Shu, Y., Gu, Y., Yasunaga, M., and Su, Y. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Proceedings of NeurIPS*, 2024.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M.-W. REALM: Retrieval-augmented language model pre-training. In *Proceedings of ICML*, 2020.

Zhang, Y., et al. HiChunk: Evaluating and enhancing hierarchical document chunking for RAG. *arXiv preprint arXiv:2509.11552*, 2025.

Li, Z., et al. Hierarchical re-ranker retriever. *arXiv preprint arXiv:2503.02401*, 2025.

Huo, S., Arabzadeh, N., and Clarke, C. L. A. Evaluation of retrieval-augmented generation: A survey. *arXiv preprint arXiv:2405.07437*, 2024.

Izacard, G. and Grave, E. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of EACL*, 2021.

Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022.

Jiang, Z., Xu, F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., and Neubig, G. Active retrieval augmented generation. In *Proceedings of EMNLP*, 2023.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*, 2020.

Khattab, O. and Zaharia, M. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of SIGIR*, 2020.

Kim, S., Suk, J., Longpre, S., Lin, B. Y., Shin, J., Welleck, S., Neubig, G., Lee, M., Lee, K., and Seo, M. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of EMNLP*, 2024.

Kocisky, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The NarrativeQA reading comprehension challenge. *Transactions of the ACL*, 6:317–328, 2018.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. Natural questions: A benchmark for question answering research. *Transactions of the ACL*, 7:453–466, 2019.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of NeurIPS*, 2020.

Li, H., et al. LLMs-as-judges: A comprehensive survey on LLM-based evaluation methods. *arXiv preprint arXiv:2412.05579*, 2024.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., and Zhu, C. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of EMNLP*, 2023.

Ni, J., Qu, C., Lu, J., Dai, Z., Hernandez Abrego, G., Ma, J., Zhao, V., Luan, Y., Hall, K., Chang, M.-W., and Yang, Y. Large dual encoders are generalizable retrievers. In *Proceedings of EMNLP*, 2022.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, 2016.

Wang, J., et al. Reconstructing context: Evaluating advanced chunking strategies for retrieval-augmented generation. *arXiv preprint arXiv:2504.19754*, 2025.

Robertson, S. and Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., and Zaharia, M. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of NAACL*, 2022.

Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., and Manning, C. D. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *Proceedings of ICLR*, 2024.

Soman, K., Rose, P. W., Morris, J. H., et al. Biomedical knowledge graph-optimized prompt generation for large language models. *Bioinformatics*, 40(9), 2024.

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., and Gurevych, I. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of NeurIPS (Datasets and Benchmarks Track)*, 2021.

VectifyAI. PageIndex: Vectorless reasoning-based RAG. `https://github.com/VectifyAI/PageIndex`, 2025.

Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. Hierarchical attention networks for document classification. In *Proceedings of NAACL*, 2016.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of EMNLP*, 2018.

Zhao, W. X., Liu, J., Ren, R., and Wen, J.-R. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems*, 2022.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Proceedings of NeurIPS*, 2023.

Zhong, Z., Liu, H., Cui, X., Zhang, X., and Qin, Z. Mix-of-Granularity: Optimize the chunking granularity for retrieval-augmented generation. In *Proceedings of COLING*, 2025.