SPROUTRAG: Attention-Guided Tree Search with Progressive Embeddings for Long-Document RAG

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) systems must balance retrieval granularity with contextual coherence, a challenge that existing methods address through LLM-guided chunking, single-level context expansion, or hierarchical summarization. These approaches variously depend on costly LLM calls during indexing or retrieval, limit context aggregation to a single granularity level, or introduce information loss through summarization. We present **SPROUTRAG**, an attention-guided hierarchical RAG framework that builds a binary tree over sentence-level chunks using learned inter-sentence attention from a sentence-level large language model (SLLM). To avoid proximity bias from uniform attention averaging, SPROUTRAG learns head- and layer-level aggregation weights that identify attention patterns most aligned with semantic document structure. At retrieval time, it uses hierarchical beam search over the tree to gather candidates at multiple granularities, capturing multi-sentence relevance beyond flat retrieval. The framework is trained end-to-end with a joint objective that improves both embeddings and tree structure, without requiring external LLM calls. Experiments across four benchmarks spanning scientific, legal, and open-domain settings demonstrate that SPROUTRAG improves information efficiency (IE) by 6.1% on average over the strongest baseline[1].

1 Introduction

Retrieval-augmented generation (RAG) has become the dominant paradigm for grounding large language models (LLMs) in external knowledge, mitigating hallucinations and enabling domain-specific reasoning (Lewis et al., 2020; Augenstein et al., 2024). As LLMs are increasingly applied to complex tasks involving long documents (Jin et al., 2025a), directly feeding entire documents as context becomes impractical due to input length limitations and degraded attention over extended sequences (Jin et al., 2025b; Liu et al., 2024). Consequently, RAG frameworks segment documents into chunks and retrieve the most relevant ones to construct focused, high-quality context.

The effectiveness of this retrieval step hinges critically on how documents are segmented. Large chunks preserve contextual coherence but introduce redundant noise that dilutes key information, while fine-grained chunks offer precision but suffer from semantic fragmentation and broken inter-chunk relationships (Tao et al., 2025; Zhao et al., 2025a). This problem is particularly acute for cross-paragraph retrieval, where answering a query requires synthesizing information scattered across multiple document sections, as in multi-hop reasoning and summarization tasks (Liu et al., 2025).

Recent work addresses this challenge from several directions. SAKI-RAG (Tao et al., 2025) uses a SLLM (An et al., 2024) to merge semantically related sentence pairs and relies on an external LLM to filter retrieval candidates. However, extending this pairwise expansion to multi-chunk relevance greatly increases the candidate space and makes LLM filtering expensive. LLM-guided chunking methods such as Meta-Chunking (Zhao et al., 2025b) and MoC (Zhao et al., 2025a) improve segmentation quality, but discard cross-chunk dependencies after chunking. Hierarchical methods such as RAPTOR (Sarthi et al., 2024) support multi-granularity retrieval through clustering and summarization, yet clustering treats chunks within a group as interchangeable and summaries can lose evidence. Graph-based methods such as GraphRAG (Edge et al., 2025) model entity relations, but are less effective when fine-grained chunks contain sparse entity information.

In this paper, we present **SPROUTRAG**, an attention-guided hierarchical RAG framework that

[1] Code and trained weights will be released publicly upon publication.

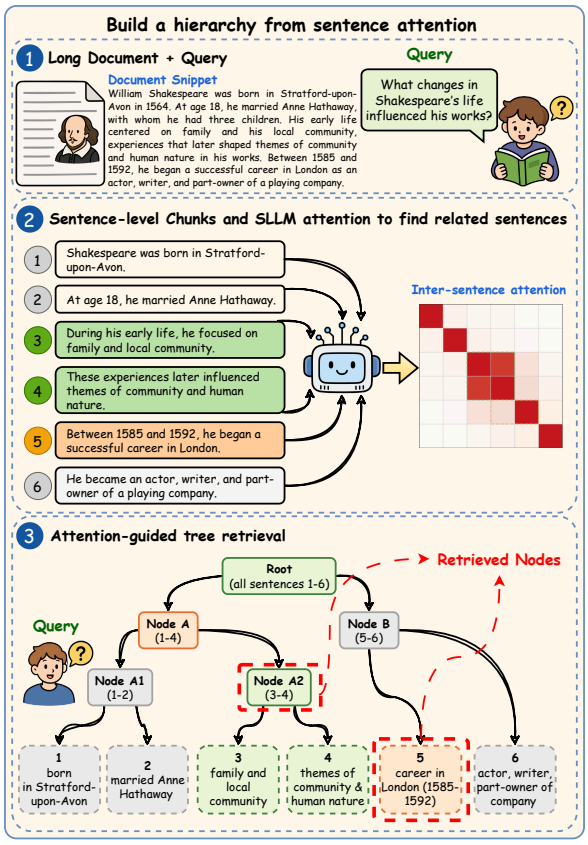

Figure 1: SPROUTRAG segments a long document into sentence-level chunks, uses SLLM attention to identify semantically related sentences, and organizes them into an attention-guided binary tree. Retrieval then selects evidence across fine-grained leaves, mid-level nodes, and broader subtrees, preserving precision while recovering coherence.

addresses the limitations of prior work without incurring LLM inference overhead at retrieval time. As illustrated in Figure 1, SPROUTRAG transforms sentence-level chunks into an attention-guided hierarchy, enabling retrieval at multiple semantic granularities. SPROUTRAG encodes documents at sentence granularity using an SLLM and constructs a binary tree bottom-up, where merge order is determined by a learned weighted aggregation of inter-sentence attention across transformer heads and layers. This aggregation replaces naive uniform averaging, which we show introduces a proximity bias that corrupts the global tree index; instead, learnable scalar weights allow the model to discover which head types best reflect semantic co-relevance for document structure. Each internal node stores a progressive embedding that compositionally represents its subtree, enabling multi-granularity retrieval via hierarchical beam search that collects candidates across all tree levels. SPROUTRAG is trained end-to-end to jointly optimize retrieval quality and tree structure, requiring no external LLM calls at any stage. It captures emergent multi-sentence relevance that pairwise or flat retrieval methods structurally cannot, while remaining efficient enough for practical deployment. We evaluate SPROUTRAG on four benchmarks spanning scientific, legal, and open-domain settings. On average, SPROUTRAG improves information efficiency (IE) by 6.1% over the strongest baseline, especially in the cases where evidence is often dispersed across paragraphs.

In summary, the contributions of this paper are as follows. **(1)** We introduce SPROUTRAG, an attention-guided hierarchical RAG framework that constructs a binary tree over sentence-level chunks using learned inter-sentence attention, enabling multi-granularity retrieval without any LLM calls at inference time. **(2)** We identify and address the proximity bias introduced by uniform attention averaging in sentence-level transformers, replacing it with a learned weighted aggregation that allows the model to discover which attention heads best reflect semantic co-relevance for document structure. **(3)** We introduce a joint training objective that simultaneously improves retrieval quality and tree structure, eliminating the need for external LLM filtering or lossy summarization at any stage of the pipeline.

2 Related Work

Chunking and adaptive retrieval. The effectiveness of retrieval-augmented generation depends strongly on how documents are segmented into retrievable units. Standard RAG pipelines often rely on rule-based splitters, such as fixed-length or delimiter-based chunking, which are efficient but insensitive to semantic boundaries (Team, 2024). Recent methods aim to improve this granularity choice. Late-Chunking (Günther et al., 2025) contextualizes token representations before forming chunk embeddings, while Meta-Chunking (Zhao et al., 2025b) and MoC (Zhao et al., 2025a) use LLM-based signals or routing mechanisms to produce more adaptive chunk boundaries. Dense X Retrieval (Chen et al., 2024) moves toward finer granularity by decomposing text into atomic propositions, improving precision but weakening broader contextual continuity. Other methods adapt retrieval after chunks have been formed. ReflectiveRAG (Verma et al., 2026) introduces a self-reflective retrieval loop that evaluates evidence suf-

ficiency and reformulates queries to improve factual grounding, but it does not change the underlying flat organization of retrieval units. Closest to our work, SAKI-RAG (Tao et al., 2025) uses a SLLM to estimate inter-sentence attention and expand retrieved chunks with related sentences. Unlike SAKI-RAG's pairwise expansion with LLM filtering, SPROUTRAG builds a global sentence-level hierarchy that supports multi-granularity retrieval without inference-time LLM calls.

Structured and hierarchical retrieval. Beyond flat chunk retrieval, structured RAG methods organize document content into higher-level representations. RAPTOR (Sarthi et al., 2024) recursively clusters chunks and summarizes each cluster into a tree, enabling retrieval from multiple levels; however, its structure is based on embedding-space clustering and relies on LLM-generated summaries, which can discard fine-grained details. Graph-based approaches such as GraphRAG (Edge et al., 2025) and LightRAG (Guo et al., 2025) represent documents through entities and relations, supporting traversal-based retrieval but depending on successful entity extraction and relation construction. PropRAG (Wang and Han, 2025) replaces entity triples with propositions and performs LLM-free beam search over proposition paths, while Beam Retrieval (Zhang et al., 2024) shows the benefit of maintaining multiple retrieval hypotheses for multi-hop passage retrieval. PageIndex (Zhang et al., 2025a) similarly explores reasoning-based, vectorless retrieval over document tree structures, but relies on document-level structural organization rather than learned sentence-level attention. SPROUTRAG instead builds an attention-guided binary tree over sentence-level chunks, with compositional internal-node representations and joint retrieval over all the nodes. This preserves cross-sentence dependencies without lossy summarization, entity-centric structure, or external LLM calls.

3 SPROUTRAG

As illustrated in Figure 2, SPROUTRAG replaces flat chunk retrieval with a **trained attention-guided hierarchy** over sentence-level chunks. During offline indexing, a SLLM encodes the document and provides both sentence embeddings and inter-sentence attention signals. These signals are aggregated with learnable head–layer weights and used to build a binary tree, where leaves represent fine-grained chunks and internal nodes store **progressive embeddings** of merged sentence groups. During online retrieval, SPROUTRAG encodes the query and performs **hierarchical beam search**, collecting candidates from leaves, internal nodes, and subtrees. The framework is trained with a joint objective that improves both **retrieval quality** and **tree structure**, enabling multi-granularity retrieval without external LLM calls during retrieval.

3.1 Attention-Guided Indexing

Given a document D, we first split it into sentence-level chunks $S = \{s_1, \ldots, s_n\}$. We encode the full sequence with a SLLM, obtaining contextualized sentence embeddings $\{e(s_i)\}_{i=1}^{n}$ and attention matrices from all layers and heads. For layer l and head h, we denote the corresponding attention matrix as $\mathrm{Attn}^{(l,h)} \in \mathbb{R}^{n\times n}$.

A uniform average over all heads and layers can overemphasize local sentence proximity, since some attention heads primarily capture sequential patterns (Voita et al., 2019). To reduce this **proximity bias**, SPROUTRAG learns a weighted aggregation over heads and layers:

$$\mathbf{A}_{ij} = \sum_{l=1}^{L}\sum_{h=1}^{H} w_{l,h}\,\mathrm{Attn}_{ij}^{(l,h)}, \tag{1}$$

where $w_{l,h}$ is defined as:

$$w_{l,h} = \frac{\exp(\alpha_{l,h})}{\sum_{l'=1}^{L}\sum_{h'=1}^{H}\exp(\alpha_{l',h'})}. \tag{2}$$

The learnable scalars $\alpha_{l,h}$ allow the model to emphasize attention heads that better capture semantic co-relevance. We then symmetrize the aggregated attention to obtain a mutual relation score:

$$\mathbf{M}_{ij} = \frac{\mathbf{A}_{ij} + \mathbf{A}_{ji}}{2}. \tag{3}$$

The tree is built bottom-up. Initially, each sentence chunk is a leaf node. At each step, we merge the pair of active nodes with the highest mutual attention score. The parent embedding is computed as a **progressive embedding** of its children:

$$e(p) = \frac{e(u) + e(v)}{2}, \tag{4}$$

where u and v are the merged child nodes. After merging, the new parent inherits its strongest relation to each remaining node:

$$\mathbf{M}_{pr} = \max(\mathbf{M}_{ur}, \mathbf{M}_{vr}). \tag{5}$$

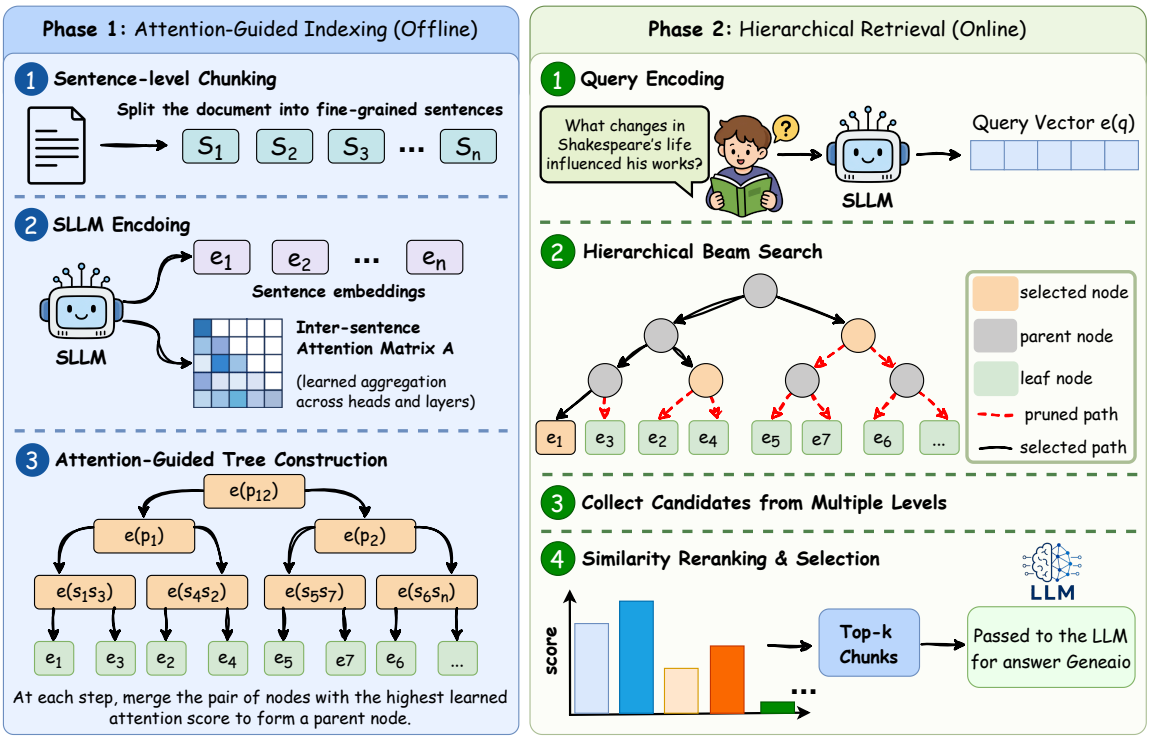

Figure 2: Overview of SPROUTRAG. In the offline indexing phase (**Phase 1**), documents are split into sentence-level chunks and encoded with a SLLM to obtain sentence embeddings and inter-sentence attention. Learned aggregation over attention heads and layers guides bottom-up tree construction, producing an attention tree with sentence embeddings at the leaves and progressive embeddings at internal nodes. In the online retrieval phase(**Phase 2**), a query is encoded and hierarchical beam search traverses the tree, collecting candidates from multiple levels before similarity reranking selects the top-k chunks for answer generation.

This single-linkage update preserves long-range semantic connections as the hierarchy grows. The result is an **attention tree** $\mathcal{T}$ whose leaves retain sentence-level precision and whose internal nodes represent broader semantic units.

3.2 Hierarchical Retrieval

Given a query q, SPROUTRAG first encodes it with the same SLLM used during indexing to obtain a query embedding $e(q)$. Retrieval is then performed over the attention-guided binary tree, where each node v represents a document span at a specific granularity. Leaf nodes correspond to sentence-level chunks, while internal nodes represent progressively larger groups of semantically related sentences. This allows SPROUTRAG to retrieve evidence at the level most appropriate for the query, rather than relying on a fixed chunk size.

Each candidate node is scored by cosine similarity between the query embedding and the node representation:

$$\mathrm{sim}(q, v) = \frac{e(q)^\top e(v)}{\|e(q)\| \, \|e(v)\|}. \tag{6}$$

Starting from the root node, retrieval proceeds via hierarchical beam search. Let $\mathcal{B}_t$ denote the active beam at depth t, with $\mathcal{B}_0 = \{v_{\text{root}}\}$. At each step, SPROUTRAG expands the children of the current beam nodes and retains the top-b most relevant nodes:

$$\mathcal{B}_{t+1} = \mathrm{Top}_b \left(\bigcup_{v \in \mathcal{B}_t} \mathrm{Child}(v), \mathrm{sim}(q, \cdot) \right), \tag{7}$$

where b is the beam width. This search strategy focuses computation on the most promising branches of the tree while still allowing the retriever to explore multiple semantically relevant regions of the document.

In parallel, SPROUTRAG collects relevant nodes encountered during traversal. Let $\mathcal{V}_{\text{visit}}$ denote the set of all nodes scored during beam search:

$$\mathcal{V}_{\text{visit}} = \bigcup_t \bigcup_{v \in \mathcal{B}_t} \mathrm{Child}(v). \tag{8}$$

The retrieval candidate set is then defined as all visited nodes whose similarity exceeds a learned threshold δ:

$$\mathcal{C} = \{ v \in \mathcal{V}_{\text{visit}} \ : \ \mathrm{sim}(q, v) \geq \delta \}. \tag{9}$$

Algorithm 1 SPROUTRAG Indexing and Retrieval

Require: Document D, query q, beam width b, threshold δ, top-k
Ensure: Retrieved evidence F

{**Offline indexing**}
$S \leftarrow \mathrm{SplitSentences}(D)$
$E, \mathcal{A} \leftarrow \mathrm{SLLM}(S)$
$\mathbf{A} \leftarrow \sum_{l,h} w_{l,h} \mathcal{A}^{(l,h)}$
$\mathbf{M} \leftarrow (\mathbf{A} + \mathbf{A}^\top)/2$
$\mathcal{N} \leftarrow \{\mathrm{Leaf}(s_i, e_i) \mid s_i \in S, e_i \in E\}$
while $|\mathcal{N}| > 1$ **do**
  $(u, v) \leftarrow \arg\max_{u \neq v;\, u,v \in \mathcal{N}} \mathbf{M}_{uv}$
  $p \leftarrow \mathrm{Node}(u, v)$
  $e_p \leftarrow (e_u + e_v)/2$
  for $r \in \mathcal{N} \setminus \{u, v\}$ **do**
    $\mathbf{M}_{pr} \leftarrow \max(\mathbf{M}_{ur}, \mathbf{M}_{vr})$
    $\mathbf{M}_{rp} \leftarrow \mathbf{M}_{pr}$
  end for
  $\mathcal{N} \leftarrow (\mathcal{N} \setminus \{u, v\}) \cup \{p\}$
end while
$\mathcal{T} \leftarrow \mathrm{root}(\mathcal{N})$
{**Online retrieval**}
$e_q \leftarrow \mathrm{SLLM}(q)$
$\mathcal{C} \leftarrow \emptyset, \quad \mathcal{B} \leftarrow \{\mathcal{T}\}$
while $\mathcal{B} \neq \emptyset$ **do**
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{v \in \mathcal{B} \mid \mathrm{sim}(e_q, e_v) \geq \delta\}$
  $\mathcal{X} \leftarrow \bigcup_{v \in \mathcal{B}} \mathrm{Children}(v)$
  $\mathcal{B} \leftarrow \mathrm{TopB}(\mathcal{X}, b, \mathrm{sim}(e_q, \cdot))$
end while
$F \leftarrow \mathrm{TopK}(\mathrm{Rerank}(\mathcal{C}, q), k)$
return F

The candidate set $\mathcal{C}$ contains evidence at multiple granularities, from sentence-level leaves to larger subtrees, allowing SPROUTRAG to retrieve either precise facts or broader multi-sentence context as needed. The collected candidates are reranked by similarity or a lightweight reranker, and the top-k chunks are passed to the answer generator. The complete indexing and retrieval procedure is summarized in Algorithm 1.

3.3 Joint Training

The pretrained SLLM is not optimized for retrieval or for constructing retrieval-oriented document structures. We therefore fine-tune SPROUTRAG with a joint objective that improves both the embedding space and the attention tree.

Retrieval objective. We train the SLLM embeddings with contrastive learning over query–passage pairs. Given a query q, a positive passage p^+, and hard negatives $\{p_j\}$, each passage is represented by mean-pooling its sentence embeddings. We optimize:

$$\mathcal{L}_{\mathrm{ret}} = -\log \frac{\exp(\mathrm{sim}(q, p^+)/\tau)}{\sum_j \exp(\mathrm{sim}(q, p_j)/\tau)}, \tag{10}$$

where τ is a temperature parameter. This objective aligns queries with relevant passages and separates them from hard negatives.

Structure objective. Good embeddings alone do not guarantee a useful hierarchy. Since the tree depends on the learned attention matrix, we add an attention regularizer that encourages co-relevant sentence pairs to receive high mutual attention. Let $\mathcal{G}$ be the set of sentence pairs within a positive passage that jointly support the query. We define:

$$\mathcal{L}_{\mathrm{attn}} = -\frac{1}{|\mathcal{G}|} \sum_{(s_i, s_j) \in \mathcal{G}} \log \left(\frac{\mathbf{A}_{ij} + \mathbf{A}_{ji}}{2} \right). \tag{11}$$

This objective directly shapes the learned head–layer aggregation so that the induced tree groups semantically related evidence into coherent subtrees.

Final objective. The final training loss is:

$$\mathcal{L} = \mathcal{L}_{\mathrm{ret}} + \lambda \mathcal{L}_{\mathrm{attn}}, \tag{12}$$

where λ controls the strength of structure regularization. After training, the learned aggregation weights are used during offline indexing, and retrieval requires only query encoding, tree traversal, and reranking. Thus, SPROUTRAG avoids external LLM filtering and lossy LLM-based summarization while supporting efficient multi-granularity retrieval.

4 Experiments and Results

4.1 Experimental Setup

Benchmarks. We evaluate SPROUTRAG on four retrieval benchmarks spanning scientific, legal, and open-domain settings: SCI-DOCS (Cohan et al., 2020), LegalBench-RAG (Pipitone and Alami, 2024), Dragonball (Zhu et al., 2025), and MS MARCO (Nguyen et al., 2016). For end-to-end answer generation, we further evaluate on HotpotQA (Yang et al., 2018), WebQuestions (Berant et al., 2013), and Dragonball. See Appendix A.1 for more details.

Baselines. We compare against representative chunking and structured retrieval methods, including Dense X Retrieval (Chen et al., 2024), Meta-Chunking (Zhao et al., 2025b), MoC (Zhao et al., 2025a), RAPTOR (Sarthi et al., 2024), LightRAG (Guo et al., 2025), PropRAG (Wang and Han, 2025), and SAKI-RAG (Tao et al., 2025). GraphRAG (Edge et al., 2025), ReflectiveRAG (Verma et al., 2026), PageIndex (Zhang et al., 2025a), and REFRAG (Lin et al., 2025) are reported only for final task performance, as they primarily involve LLM-heavy reasoning, generation, summarization, or decoding-time optimization rather than efficient retrieval. For fair comparison, methods requiring an LLM or reranker use the same QWEN/QWEN3-8B (Team, 2025) generator and QWEN/QWEN3-RERANKER-4B (Zhang et al., 2025b) reranker. Other than unifying the generator and reranker, we follow the settings recommended in the original papers for each baseline. See Appendix A.2 for more details.

Metrics. For retrieval evaluation, we report Recall, Precision, and Information Efficiency (IE), where $\text{IE} = \text{Recall} \times \text{Precision}$. We compute each metric at $k \in \{1, 3, 5\}$ and report the average across these three cutoffs. For end-to-end generation, we report F1 on HotpotQA and WebQuestions, and ROUGE-L (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and BERTScore (Zhang* et al., 2020) on Dragonball. We also report online efficiency using Tok/Q and latency, where Tok/Q counts online model-token usage per query, excluding offline indexing and output tokens. All reported results are averaged over three independent runs.

Implementation and Training Details. We use the 1.3B-parameter SLLM (An et al., 2024)[2] as the sentence encoder and split documents into chunks of up to two sentences. SPROUTRAG is trained on 30K query–passage examples sampled from CLaRa (He et al., 2025), fine-tuning the SLLM and learned head–layer aggregation weights with the joint objective in Eq. 12. Unless otherwise stated, we use the following hyperparameters as the default setting: 3 training epochs with AdamW, learning rates of 2×10^{-5} for the SLLM and 1×10^{-3} for the aggregation scalars, batch size 32, temperature $\tau = 0.05$, attention weight $\lambda = 0.1$, and 5% linear warmup. At retrieval time, the default setting uses beam width $b = 5$, collects candidates from all tree levels, and reranks them with QWEN3-RERANKER-4B. Final answers are generated with QWEN3-8B. For fair comparison, baselines that require an LLM or reranker use the same models; otherwise, we follow their original settings. All experiments use 8 NVIDIA A100 80GB GPUs, and results are averaged over three runs.

[2] https://github.com/cavedweller509/SentenceVAE

4.2 Retrieval Quality

Table 1 reports retrieval performance across four benchmarks. SPROUTRAG achieves the highest IE on all datasets, improving over the strongest baseline by **8.06** points on Dragonball, **4.65** on SCIDOCS, **4.90** on LegalBench-RAG, and **6.83** on MS MARCO. These improvements are not driven by recall alone: SPROUTRAG also obtains the **best precision on every benchmark**. This suggests that the attention-guided hierarchy helps retrieve broader supporting context while avoiding the noise introduced by overly large or weakly related chunks.

The comparison with SAKI-RAG is especially informative. While SAKI-RAG achieves strong precision, particularly on Dragonball and SCIDOCS, its pairwise expansion limits evidence aggregation, reducing recall and IE. In contrast, SPROUTRAG converts sentence-level attention into a **global tree structure**, enabling retrieval over individual chunks, internal nodes, and subtrees. This preserves SAKI-RAG's precision benefits while improving IE across all datasets. Structured baselines such as RAPTOR, LightRAG, PropRAG, and MoC improve recall over flat or boundary-based chunking, but their clustering, graph, proposition, or routing structures do not explicitly model learned multi-sentence composition. SPROUTRAG bridges this gap: leaves retain fine-grained evidence, while internal nodes recover coherent context, yielding the strongest recall–precision tradeoff. Appendix C provides a qualitative example.

4.3 End-to-End Performance and Efficiency

We next evaluate whether the retrieval improvements translate into stronger final answers. Table 2 compares SPROUTRAG with system-level RAG methods on HotpotQA, WebQuestions, and Dragonball. PageIndex achieves the highest final answer scores, but it requires substantially more online computation due to its reasoning-based search and evidence construction. REFRAG improves efficiency compared with reflection-heavy or reasoning-heavy systems, but SPROUTRAG still provides the strongest **performance–efficiency tradeoff**: it outperforms

Method	Dragonball			SCI-DOCS		
	Rec. ↑	Pre. ↑	IE ↑	Rec. ↑	Pre. ↑	IE ↑
Dense X Retrieval	$3.18_{\pm 0.09}$	$5.76_{\pm 0.14}$	$0.18_{\pm 0.02}$	$95.64_{\pm 0.31}$	$88.13_{\pm 0.42}$	$84.31_{\pm 0.47}$
Meta-Chunking-PPL	$42.91_{\pm 0.54}$	$44.62_{\pm 0.63}$	$19.14_{\pm 0.37}$	$22.16_{\pm 0.71}$	$18.73_{\pm 0.64}$	$4.15_{\pm 0.19}$
Meta-Chunking-MSP	$32.74_{\pm 0.47}$	$42.08_{\pm 0.58}$	$13.77_{\pm 0.31}$	$96.37_{\pm 0.29}$	$88.42_{\pm 0.46}$	$85.21_{\pm 0.43}$
RAPTOR	$32.19_{\pm 0.61}$	$41.92_{\pm 0.76}$	$13.49_{\pm 0.34}$	$97.72_{\pm 0.24}$	$93.03_{\pm 0.39}$	$90.92_{\pm 0.41}$
LightRAG	$47.38_{\pm 0.69}$	$54.16_{\pm 0.72}$	$25.67_{\pm 0.46}$	$97.51_{\pm 0.27}$	$92.84_{\pm 0.44}$	$90.53_{\pm 0.45}$
PropRAG	$49.81_{\pm 0.64}$	$58.24_{\pm 0.79}$	$29.01_{\pm 0.52}$	$98.07_{\pm 0.22}$	$94.36_{\pm 0.35}$	$92.55_{\pm 0.38}$
MoC	$51.26_{\pm 0.57}$	$60.47_{\pm 0.68}$	$30.99_{\pm 0.44}$	$98.18_{\pm 0.19}$	$94.73_{\pm 0.31}$	$93.01_{\pm 0.34}$
SAKI-RAG	$36.83_{\pm 0.52}$	$78.61_{\pm 0.64}$	$28.95_{\pm 0.41}$	$91.82_{\pm 0.33}$	$97.43_{\pm 0.28}$	$89.46_{\pm 0.36}$
SproutRAG	$\mathbf{45.76}_{\pm 0.43}$	$\mathbf{85.34}_{\pm 0.51}$	$\mathbf{39.05}_{\pm 0.38}$	$\mathbf{98.74}_{\pm 0.16}$	$\mathbf{98.91}_{\pm 0.18}$	$\mathbf{97.66}_{\pm 0.21}$

Method	LegalBench-RAG			MS MARCO		
	Rec. ↑	Pre. ↑	IE ↑	Rec. ↑	Pre. ↑	IE ↑
Dense X Retrieval	$24.83_{\pm 0.38}$	$35.74_{\pm 0.51}$	$8.88_{\pm 0.21}$	$58.41_{\pm 0.46}$	$62.68_{\pm 0.55}$	$36.61_{\pm 0.48}$
Meta-Chunking-PPL	$27.82_{\pm 0.44}$	$38.41_{\pm 0.57}$	$10.69_{\pm 0.26}$	$60.84_{\pm 0.52}$	$63.97_{\pm 0.49}$	$38.92_{\pm 0.45}$
Meta-Chunking-MSP	$28.31_{\pm 0.41}$	$39.26_{\pm 0.49}$	$11.12_{\pm 0.24}$	$61.73_{\pm 0.43}$	$64.36_{\pm 0.52}$	$39.73_{\pm 0.39}$
RAPTOR	$28.64_{\pm 0.48}$	$39.83_{\pm 0.62}$	$11.41_{\pm 0.29}$	$62.48_{\pm 0.56}$	$65.81_{\pm 0.67}$	$41.12_{\pm 0.53}$
LightRAG	$30.62_{\pm 0.53}$	$43.97_{\pm 0.58}$	$13.47_{\pm 0.33}$	$66.13_{\pm 0.51}$	$68.42_{\pm 0.61}$	$45.25_{\pm 0.57}$
PropRAG	$31.47_{\pm 0.46}$	$45.28_{\pm 0.54}$	$14.25_{\pm 0.32}$	$66.92_{\pm 0.48}$	$69.83_{\pm 0.58}$	$46.73_{\pm 0.49}$
MoC	$32.18_{\pm 0.42}$	$46.11_{\pm 0.51}$	$14.84_{\pm 0.27}$	$67.31_{\pm 0.45}$	$70.26_{\pm 0.53}$	$47.29_{\pm 0.46}$
SAKI-RAG	$31.38_{\pm 0.39}$	$46.27_{\pm 0.47}$	$14.52_{\pm 0.25}$	$68.04_{\pm 0.42}$	$71.58_{\pm 0.49}$	$48.70_{\pm 0.44}$
SproutRAG	$\mathbf{36.91}_{\pm 0.34}$	$\mathbf{53.48}_{\pm 0.41}$	$\mathbf{19.74}_{\pm 0.29}$	$\mathbf{72.86}_{\pm 0.37}$	$\mathbf{76.21}_{\pm 0.44}$	$\mathbf{55.53}_{\pm 0.40}$

Table 1: Retrieval performance across four benchmarks. Recall, Precision, and IE are averaged over @1, @3, and @5, with IE computed at each cutoff before averaging. Values report the mean over three independent runs, and the red ± values indicate the corresponding standard deviation. The shaded row marks SproutRAG. Refer to Appendix B for the results @1, @3, and @5.

GraphRAG, ReflectiveRAG, and REFRAG across all final-performance metrics, while using only **4.38K online tokens per query** and **193 ms** latency. The reported cost measures online per-query inference and excludes offline training and indexing. **SproutRAG does require an upfront training stage**: we fine-tune the SLLM and attention aggregation weights on a 30K-example subset of CLaRa. However, **this cost is paid once and reused across datasets**, similar to other systems with offline preparation or model adaptation costs. The cross-dataset results show that the learned attention-guided hierarchy generalizes without retraining for each benchmark, making the training cost amortized rather than query-time overhead.

4.4 Ablation Study

Training objectives. Table 3, in the first group, evaluates the role of the training objectives. The *Not trained* variant performs worst across all datasets, showing that the pretrained SLLM attention and embeddings are not sufficient for retrieval-oriented tree construction. Removing $\mathcal{L}_{\text{ret}}$ substantially reduces both recall and IE, since the query and evidence embeddings are no longer explicitly aligned. Removing $\mathcal{L}_{\text{attn}}$ is less damaging than removing the retrieval loss, but still causes a consistent drop, especially in IE. This confirms that **embedding quality and tree quality require complementary supervision**: the retrieval loss aligns query–passage representations, while the attention-structure loss shapes the hierarchy used for multi-granularity retrieval.

Tree and retrieval design. The second group in Table 3 examines the necessity of the attention-guided hierarchy. Uniform attention aggregation reduces performance, highlighting that averaging heads and layers introduces **proximity bias** and weakens the tree. An embedding-similarity tree also underperforms, showing that SLLM attention encodes structural information beyond embeddings. Leaf-only retrieval maintains high precision but lowers recall and IE, while greedy search suffers from early path commitment. These results demonstrate that **learned attention aggregation**, **internal-node retrieval**, and **beam search** are all crucial for balancing precise evidence with broader contextual coverage.

Hyperparameter sensitivity. The final group in Table 3 studies beam width b and attention regu-

Method	Final Performance					Average Cost	
	HotpotQA	WebQuestions	Dragonball				
	F1 ↑	F1 ↑	R-L ↑	MTR ↑	BRT ↑	Tok/Q ↓	Lat. (ms) ↓
GraphRAG	72.18	64.73	0.346	0.361	0.637	16238	2317
ReflectiveRAG	70.64	63.29	0.334	0.349	0.622	11274	1186
REFRAG	73.42	65.38	0.351	0.368	0.641	5436	492
PageIndex	**79.36**	**70.81**	**0.389**	**0.406**	**0.691**	24620	2847
SPROUTRAG	76.47	68.12	0.372	0.389	0.671	**4382**	**193**

Table 2: End-to-end answer quality and online efficiency. HotpotQA and WebQuestions are evaluated with F1, while Dragonball uses ROUGE-L (R-L), METEOR (MTR), and BERTScore (BRT). Tok/Q counts online model input tokens per query, excluding offline training, indexing, and output tokens; Lat. reports online per-query latency. All methods use the same generator and reranker when applicable.

Variant	Dragonball			SCI-DOCS			LegalBench-RAG			MS MARCO		
	Rec. ↑	Pre. ↑	IE ↑	Rec. ↑	Pre. ↑	IE ↑	Rec. ↑	Pre. ↑	IE ↑	Rec. ↑	Pre. ↑	IE ↑
SPROUTRAG ($b = 5, \lambda = 0.1$)	**45.76**	**85.34**	**39.05**	**98.74**	**98.91**	**97.66**	**36.91**	**53.48**	**19.74**	**72.86**	**76.21**	**55.53**
Training Objectives												
Not trained	34.28	69.41	23.79	90.36	92.18	83.28	29.47	42.53	12.53	64.18	67.42	43.28
w/o $\mathcal{L}_{\text{ret}}$	37.64	73.52	27.67	92.71	94.86	87.96	31.08	45.17	14.04	66.24	69.73	46.18
w/o $\mathcal{L}_{\text{attn}}$	41.39	78.26	32.39	96.42	96.88	93.41	33.52	48.31	16.20	69.18	72.64	50.25
Tree and Retrieval Design												
Uniform attention aggregation	39.18	76.94	30.15	95.83	96.12	92.11	32.74	47.36	15.51	68.41	71.52	48.92
Embedding-similarity tree	40.72	79.38	32.32	96.31	96.74	93.17	33.18	48.42	16.06	69.36	72.48	50.28
Leaf-only retrieval	38.26	83.19	31.83	94.87	98.12	93.07	31.94	52.61	16.80	67.83	75.42	51.16
Greedy search	39.84	82.47	32.86	96.25	98.34	94.65	32.81	51.76	16.98	68.92	74.86	51.60
Hyperparameter Sensitivity												
Beam width $b = 3$	44.12	84.71	37.37	98.31	98.83	97.16	35.72	52.86	18.88	71.94	75.63	54.42
Beam width $b = 10$	46.24	82.63	38.21	98.91	98.42	97.35	37.28	51.92	19.36	73.41	74.32	54.56
$\lambda = 0.05$	44.68	83.92	37.50	98.42	98.63	97.07	35.96	52.47	18.87	72.14	75.48	54.46
$\lambda = 0.20$	45.18	82.74	37.38	98.58	98.37	96.97	36.42	51.83	18.88	72.49	74.93	54.31

Table 3: Ablation study on retrieval performance. Metrics are averaged over @1, @3, and @5. The blue row is the default SPROUTRAG setting ($b = 5, \lambda = 0.1$). The three groups evaluate training objectives, tree/retrieval design, and sensitivity to b and λ.

larization weight λ. Reducing the beam width to $b = 3$ slightly lowers IE because fewer semantic paths are explored, while increasing it to $b = 10$ improves recall but slightly reduces precision, yielding no overall advantage over the default $b = 5$. Similarly, both $\lambda = 0.05$ and $\lambda = 0.20$ underperform the default $\lambda = 0.1$: a weaker structure loss provides insufficient guidance for tree construction, while a stronger one can overemphasize attention alignment at the expense of retrieval precision. Overall, SPROUTRAG is stable across reasonable settings, with $b = 5$ and $\lambda = 0.1$ providing the best recall–precision tradeoff.

5 Conclusion and Future Work

We introduced SPROUTRAG, an attention-guided hierarchical RAG framework that organizes sentence-level chunks into a learned tree for multi-granularity retrieval. Rather than relying on fixed chunk boundaries, pairwise context expansion, lossy summarization, or inference-time LLM filtering, SPROUTRAG uses learned SLLM attention aggregation to construct a retrieval-oriented hierarchy. At inference time, hierarchical beam search selects evidence from sentence leaves, internal nodes, and broader subtrees, allowing the retriever to balance fine-grained precision with contextual coherence. Across benchmarks, SPROUTRAG improves retrieval information efficiency by 6.1% on average, offering a strong **performance–efficiency tradeoff** that approaches LLM-heavy systems while using far fewer online tokens and lower latency. While SPROUTRAG generalizes well after one-time training, several directions remain open. Future work can explore richer node composition functions beyond mean pooling, such as gated or attention-based composition, and dynamic tree adaptation or query-dependent traversal policies for complex multi-hop retrieval.

Limitations

While SPROUTRAG improves multi-granularity retrieval without inference-time LLM filtering, it has some limitations. First, the hierarchy is currently built as a **binary tree**, which may be restrictive when several sentences jointly form a coherent semantic unit and should be grouped together simultaneously. Multi-branch trees could better capture such many-to-many dependencies. Second, SPROUTRAG requires an upfront training stage for the SLLM and attention aggregation weights. Although this is a one-time cost that transfers across datasets in our experiments, it is still more expensive than using an off-the-shelf retriever without adaptation. Finally, tree construction is performed offline and remains fixed during retrieval; this makes inference efficient, but it may be less flexible for queries that require reorganizing evidence according to query-specific notions of relevance.

Ethical Considerations

Potential Risks. SPROUTRAG is designed to improve retrieval quality and reduce unnecessary inference-time LLM calls, but it does not eliminate the risks associated with RAG systems. Retrieved evidence may still be incomplete, outdated, or incorrectly interpreted by the generator, especially in high-stakes domains such as legal, medical, or financial settings. In such cases, SPROUTRAG should be used as an assistive retrieval component rather than a substitute for expert review. In addition, stronger retrieval over large document collections may surface sensitive or private information if the underlying corpus is not properly governed. Deployments should therefore include appropriate access control, data filtering, logging policies, and human oversight.

Use of AI Assistance. During the preparation of this manuscript, we used large language models for paraphrasing and improving the writing quality of the paper, and for generating icon images used in figures. All ideas, technical contributions, experimental design, and conclusions are entirely original and the work of the authors.

Reproducibility Statement. We describe the experimental setup, datasets, metrics, implementation details, and training hyperparameters in Section 4.1, with additional dataset and baseline details provided in Appendix A.1 and Appendix A.2. Upon publication, we will release our code and checkpoints under a permissive open-source license. All external datasets and models are used according to their original licenses and terms of use. All benchmarks, datasets, and baseline artifacts used in this paper are publicly available, and our use of them is limited to research evaluation, consistent with their intended use and original access conditions. Our released code and checkpoints are intended for research purposes and for reproducing the experiments in this paper.

Use of Existing Artifacts and Data. We use only publicly available benchmarks, datasets, models, and baseline artifacts, and our use is limited to research evaluation. We do not collect new user data or attempt to identify individuals. Some source corpora may contain naturally occurring named entities or web-derived text from the original datasets; we follow the datasets' original licenses, preprocessing, and access conditions. All benchmarks, datasets, and baseline methods used in this paper are publicly available, and our released code and checkpoints are intended for research and reproducibility purposes.

References

Hongjun An, Yifan Chen, Zhe Sun, and Xuelong Li. 2024. Sentencevae: Enable next-sentence prediction for large language models with faster speed, higher accuracy and longer context. *Preprint*, arXiv:2408.00655.

Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, and 1 others. 2024. Factuality challenges in the era of large language models and opportunities for fact-checking. *Nature Machine Intelligence*, 6(8):852–863.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177, Miami, Florida, USA. Association for Computational Linguistics.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online. Association for Computational Linguistics.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From local to global: A graph rag approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761, Suzhou, China. Association for Computational Linguistics.

Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao. 2025. Late chunking: Contextual chunk embeddings using long-context embedding models. *Preprint*, arXiv:2409.04701.

Jie He, Richard He Bai, Sinead Williamson, Jeff Z. Pan, Navdeep Jaitly, and Yizhe Zhang. 2025. Clara: Bridging retrieval and generation with continuous latent reasoning. *Preprint*, arXiv:2511.18659.

Bowen Jin, Jinsung Yoon, Jiawei Han, and Sercan O Arik. 2025a. Long-context LLMs meet RAG: Overcoming challenges for long inputs in RAG. In *The Thirteenth International Conference on Learning Representations*.

Bowen Jin, Jinsung Yoon, Jiawei Han, and Sercan O Arik. 2025b. Long-context LLMs meet RAG: Overcoming challenges for long inputs in RAG. In *The Thirteenth International Conference on Learning Representations*.

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

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Xiaoqiang Lin, Aritra Ghosh, Bryan Kian Hsiang Low, Anshumali Shrivastava, and Vijai Mohan. 2025. Refrag: Rethinking rag based decoding. *Preprint*, arXiv:2509.01092.

Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. 2025. HopRAG: Multi-hop reasoning for logic-aware retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 1897–1913, Vienna, Austria. Association for Computational Linguistics.

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

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Nicholas Pipitone and Ghita Houir Alami. 2024. Legalbench-rag: A benchmark for retrieval-augmented generation in the legal domain. *Preprint*, arXiv:2408.10343.

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

Wenyu Tao, Xiaofen Xing, Zeliang Li, and Xiangmin Xu. 2025. SAKI-RAG: Mitigating context fragmentation in long-document RAG via sentence-level attention knowledge integration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 1195–1213, Suzhou, China. Association for Computational Linguistics.

LangChain Team. 2024. Langchain: A framework for developing applications powered by language models.

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

Akshay Verma, Swapnil Gupta, Siddharth Pillai, Prateek Sircar, and Deepak Gupta. 2026. ReflectiveRAG: Rethinking adaptivity in retrieval-augmented generation. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 377–384, Rabat, Morocco. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Jingjin Wang and Jiawei Han. 2025. PropRAG: Guiding retrieval with beam search over proposition paths. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6212–6227, Suzhou, China. Association for Computational Linguistics.

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

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Liu Yong, and Shen Huang. 2024. End-to-end beam retrieval for multi-hop question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1718–1731, Mexico City, Mexico. Association for Computational Linguistics.

Mingtian Zhang, Yu Tang, and PageIndex Team. 2025a. Pageindex: Next-generation vectorless, reasoning-based rag. *PageIndex Blog*.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

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

Jihao Zhao, Zhiyuan Ji, Zhaoxin Fan, Hanyu Wang, Simin Niu, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2025a. MoC: Mixtures of text chunking learners for retrieval-augmented generation system. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5172–5189, Vienna, Austria. Association for Computational Linguistics.

Jihao Zhao, Zhiyuan Ji, Yuchen Feng, Pengnian Qi, Simin Niu, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2025b. Meta-chunking: Learning text segmentation and semantic completion via logical perception. *Preprint*, arXiv:2410.12788.

Kunlun Zhu, Yifan Luo, Dingling Xu, Yukun Yan, Zhenghao Liu, Shi Yu, Ruobing Wang, Shuo Wang, Yishan Li, Nan Zhang, Xu Han, Zhiyuan Liu, and Maosong Sun. 2025. RAGEval: Scenario specific RAG evaluation dataset generation framework. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8520–8544, Vienna, Austria. Association for Computational Linguistics.

A Benchmark and Baseline Details

A.1 Benchmark Dataset Details

We include four retrieval-focused benchmarks—SCI-DOCS, LegalBench-RAG, Dragonball, and MS MARCO—and three end-to-end generation benchmarks—HotpotQA, WebQuestions, and Dragonball. Together, these datasets cover scientific retrieval, legal retrieval, open-domain passage retrieval, multi-hop question answering, short-form factual QA, and multi-domain RAG evaluation.

SCI-DOCS (Cohan et al., 2020). SCI-DOCS is a scientific document representation benchmark introduced with SPECTER. It contains multiple document-level tasks, including citation prediction, document classification, and recommendation. We use SCI-DOCS as a scientific retrieval benchmark because scientific abstracts are dense, terminology-heavy, and often contain multiple related concepts within a short span. This makes the dataset useful for testing whether SPROUTRAG can construct coherent sentence-level hierarchies in technical domains.

LegalBench-RAG (Pipitone and Alami, 2024). LegalBench-RAG is designed specifically to evaluate retrieval in legal RAG pipelines. It contains 6,858 query-answer pairs over legal documents such as NDAs, M&A agreements, commercial contracts, and privacy policies. Unlike broad document retrieval, LegalBench-RAG emphasizes precise snippet retrieval: the model must identify minimal legal evidence rather than simply retrieve a generally relevant document. This makes it a strong test of fine-grained precision.

Dragonball (Zhu et al., 2025). Dragonball is a multi-domain and multilingual RAG benchmark released as part of RAGEval. It contains questions across finance, legal, and medical scenarios in English and Chinese. We use Dragonball for both retrieval and end-to-end generation because it combines heterogeneous domains, long evidence contexts, and domain-specific terminology. This setting tests whether retrieval methods can recover relevant evidence without introducing excessive distractor context.

MS MARCO (Nguyen et al., 2016). MS MARCO is a large-scale open-domain passage retrieval benchmark built from real web search queries. Its corpus consists of millions of short

passages, and the task requires identifying passages that answer natural language questions. Compared with SCI-DOCS and LegalBench-RAG, MS MARCO has shorter retrieval units and more direct query-passage matching, providing a complementary test of retrieval effectiveness when evidence is already compact.

HotpotQA (Yang et al., 2018). HotpotQA is an open-domain multi-hop QA benchmark built from Wikipedia. Its questions require reasoning over multiple supporting documents or facts, and the dataset provides sentence-level supporting-fact annotations. We use HotpotQA for end-to-end answer generation because it directly tests whether retrieved evidence supports multi-step reasoning, which aligns with SPROUTRAG's goal of retrieving evidence across multiple granularities.

WebQuestions (Berant et al., 2013). WebQuestions is an open-domain factual QA benchmark built from natural language questions collected from web search logs. The answers are typically short entities or phrases, making token-level F1 a suitable evaluation metric. We include WebQuestions to evaluate whether SPROUTRAG also improves short-form factual QA, where retrieval must remain precise and avoid adding unnecessary context.

A.2 Baseline Details

We compare SPROUTRAG against two groups of baselines: efficient retrieval-oriented methods used in the retrieval evaluation, and system-level RAG methods used in the end-to-end generation comparison. Unless otherwise stated, we follow the configurations recommended in the original papers. For methods requiring an LLM generator or reranker, we use the same QWEN3-8B generator and QWEN3-RERANKER-4B reranker for fair comparison.

Dense X Retrieval (Chen et al., 2024). Dense X Retrieval decomposes documents into fine-grained propositions and uses these propositions as retrieval units. This improves precision by making each unit more atomic and self-contained. However, proposition-level retrieval can weaken broader contextual continuity, since related facts are retrieved independently rather than as coherent multi-sentence evidence.

Meta-Chunking (Zhao et al., 2025b). Meta-Chunking uses LLM-based signals to identify semantically meaningful chunk boundaries instead of relying on fixed-size segmentation. We evaluate both variants: *Meta-Chunking-PPL*, which uses perplexity changes to detect boundaries, and *Meta-Chunking-MSP*, which uses margin-sampling-based boundary decisions. These methods improve chunk coherence, but once chunks are formed, cross-chunk semantic dependencies are not explicitly modeled.

MoC (Zhao et al., 2025a). MoC improves over single-strategy chunking by dynamically routing text to different chunking strategies or granularity choices. It is a strong adaptive chunking baseline because it can better match the segmentation strategy to the local document structure. However, MoC still primarily operates at the chunk-construction stage and does not build a retrieval-time hierarchy over sentence-level evidence.

RAPTOR (Sarthi et al., 2024). RAPTOR recursively clusters chunks and summarizes each cluster to build a hierarchical tree. Retrieval can then operate over both lower-level chunks and higher-level summaries. This provides a natural multi-granularity baseline, but its hierarchy is based on embedding-space clustering and LLM-generated summaries, which can introduce information loss and additional indexing cost.

LightRAG (Guo et al., 2025). LightRAG augments retrieval with graph-structured knowledge and combines local and global retrieval signals. It is designed to improve retrieval over connected evidence by exploiting entity and relation structure. We include it as a structured retrieval baseline, especially for settings where graph-style evidence organization can improve coverage.

PropRAG (Wang and Han, 2025). PropRAG represents documents using propositions and performs beam-style traversal over proposition paths. It is closely related to our use of beam search, but differs in its underlying structure: PropRAG searches over a proposition graph, whereas SPROUTRAG searches over an attention-guided sentence hierarchy. PropRAG is therefore a strong baseline for testing whether hierarchical sentence-level structure provides benefits beyond proposition-path retrieval.

SAKI-RAG (Tao et al., 2025). SAKI-RAG uses a Sentence-Level Large Language Model (SLLM) to estimate inter-sentence attention and expand

retrieved chunks with related sentences. It is the closest baseline to SPROUTRAG because it also uses sentence-level attention signals. However, SAKI-RAG performs pairwise expansion and relies on LLM filtering during retrieval, while SPROUTRAG converts learned attention into a global tree and retrieves across multiple granularities without inference-time LLM filtering.

GraphRAG (Edge et al., 2025). GraphRAG constructs an entity-relation graph from the corpus and uses graph structure to support retrieval and generation. It is effective for queries that align well with entity-centric evidence, but it depends on reliable entity extraction and relation construction. We include GraphRAG in the end-to-end comparison because it is a system-level RAG method with substantial LLM-based preprocessing and reasoning.

ReflectiveRAG (Verma et al., 2026). ReflectiveRAG introduces adaptive retrieval and generation through self-reflection. Instead of using a fixed retrieval budget, it evaluates whether retrieved evidence is sufficient and can reformulate or expand retrieval when needed. This makes it a strong final-performance baseline, but its main contribution lies in adaptive evidence use and generation rather than efficient retrieval structure.

PageIndex (Zhang et al., 2025a). PageIndex replaces conventional vector retrieval with a reasoning-based hierarchical page index. An LLM navigates the index and selects evidence through multi-step reasoning, which can improve final answer quality. However, because it performs LLM-heavy online search and evidence construction, we include it only in end-to-end performance comparisons rather than as an efficient retrieval baseline.

REFRAG (Lin et al., 2025). REFRAG focuses on generation-side efficiency for RAG by exploiting sparsity in retrieved contexts. It compresses retrieved chunks and selectively expands them during decoding, reducing the effective context processed by the generator. Since its main contribution is decoding-time optimization rather than retrieval or indexing, we include it in the final-performance and efficiency comparison.

B Retrieval Performance at Different Cutoffs

Tables 4, 5, and 6 report retrieval performance at cutoffs k=1, k=3, and k=5, respectively, across all six benchmarks. As expected, both precision and recall increase monotonically with k for all methods, since retrieving more documents provides greater coverage of relevant passages. The relative ordering of methods remains consistent across all cutoffs: SPROUTRAG outperforms all non-oracle baselines at every depth. This consistency demonstrates that the gains from topic-guided retrieval are not specific to any particular cutoff, but reflect a robust improvement in retrieval quality across the full range of evaluation settings reported here.

C Qualitative Analysis: Recovering Multi-Sentence Legal Evidence

Table 7 presents a qualitative example that illustrates why retrieving **internal tree nodes** is useful. The query asks whether a software services agreement limits the provider's liability and what exceptions apply. A complete answer requires more than a single sentence or a pair of related sentences: the model must recover the excluded damages, the aggregate liability cap, the scope of the limitation across legal theories, and the carve-outs for indemnification, confidentiality breaches, gross negligence, and willful misconduct. These pieces form a coherent clause-level unit, but they are distributed across four sentences. MoC retrieves a locally coherent chunk containing the liability cap, but this evidence is too narrow to answer the full query. SAKI-RAG improves over local chunking by linking the damage-exclusion sentence with the liability-cap sentence; however, its pairwise expansion still misses the later exception sentence, which is essential for a legally complete answer. In contrast, SPROUTRAG retrieves the internal node $v_{1:4}$, which groups all four relevant sentences into a single clause-level unit. This allows the generator to answer both parts of the query: the agreement limits liability through damage exclusions and a monetary cap, but the limitation does not apply to the specified carve-outs.

Method	LegalBench-RAG			MSMARCO			SCI-DOCS			Dragonball		
	IE↑	*Prec.*↑	*Rec.*↑	*IE*↑	*Prec.*↑	*Rec.*↑	*IE*↑	*Prec.*↑	*Rec.*↑	*IE*↑	*Prec.*↑	*Rec.*↑
DenseXRetrieval	1.93	32.74	5.88	19.14	58.68	32.61	70.89	86.13	82.31	0.01	1.64	1.12
Meta-Chunking-PPL	2.72	35.41	7.69	20.94	59.97	34.92	0.36	16.73	2.15	6.72	41.62	16.14
Meta-Chunking-MSP	2.94	36.26	8.12	21.57	60.36	35.73	71.91	86.42	83.21	4.21	39.08	10.77
RAPTOR	3.10	36.83	8.41	22.94	61.81	37.12	80.94	91.03	88.92	4.08	38.92	10.49
LightRAG	4.29	40.97	10.47	26.57	64.42	41.25	80.42	90.84	88.53	11.60	51.16	22.67
PropRAG	4.76	42.28	11.25	28.13	65.83	42.73	83.63	92.36	90.55	14.37	55.24	26.01
MoC	5.10	43.11	11.84	28.68	66.26	43.29	84.39	92.73	91.01	16.09	57.47	27.99
SAKI-RAG	4.98	43.27	11.52	30.21	67.58	44.70	83.46	95.43	87.46	19.62	75.61	25.95
SPROUTRAG	**8.45**	**50.48**	**16.74**	**37.21**	**72.21**	**51.53**	**93.91**	**98.04**	**95.79**	**29.68**	**82.34**	**36.05**

Table 4: Retrieval performance at depth k=1 across four benchmarks (IE↑, Precision↑, Recall↑). **Bold** = best; underline = second-best. SPROUTRAG rows are shaded.

Method	LegalBench-RAG			MSMARCO			SCI-DOCS			Dragonball		
	IE↑	*Prec.*↑	*Rec.*↑	*IE*↑	*Prec.*↑	*Rec.*↑	*IE*↑	*Prec.*↑	*Rec.*↑	*IE*↑	*Prec.*↑	*Rec.*↑
DenseXRetrieval	2.84	34.99	8.13	21.96	61.68	35.61	73.44	87.63	83.81	0.01	5.01	0.14
Meta-Chunking-PPL	3.74	37.66	9.94	23.88	62.97	37.92	0.67	18.23	3.65	8.07	43.87	18.39
Meta-Chunking-MSP	3.99	38.51	10.37	24.54	63.36	38.73	74.48	87.92	84.71	5.38	41.33	13.02
RAPTOR	4.17	39.08	10.66	26.00	64.81	40.12	83.67	92.53	90.42	5.25	41.17	12.74
LightRAG	5.50	43.22	12.72	29.83	67.42	44.25	83.13	92.34	90.03	13.31	53.41	24.92
PropRAG	6.01	44.53	13.50	31.48	68.83	45.73	86.40	93.86	92.05	16.25	57.49	28.26
MoC	6.39	45.36	14.09	32.06	69.26	46.29	87.17	94.23	92.51	18.06	59.72	30.24
SAKI-RAG	6.27	45.52	13.77	33.67	70.58	47.70	86.23	96.93	88.96	21.96	77.86	28.20
SPROUTRAG	**10.01**	**52.73**	**18.99**	**41.01**	**75.21**	**54.53**	**95.92**	**98.69**	**97.19**	**32.40**	**84.59**	**38.30**

Table 5: Retrieval performance at depth k=3 across four benchmarks (IE↑, Precision↑, Recall↑). **Bold** = best; underline = second-best. SPROUTRAG rows are shaded.

Method	LegalBench-RAG			MSMARCO			SCI-DOCS			Dragonball		
	IE↑	*Prec.*↑	*Rec.*↑	*IE*↑	*Prec.*↑	*Rec.*↑	*IE*↑	*Prec.*↑	*Rec.*↑	*IE*↑	*Prec.*↑	*Rec.*↑
DenseXRetrieval	4.99	39.49	12.63	28.16	67.68	41.61	78.68	90.63	86.81	0.04	9.51	0.40
Meta-Chunking-PPL	6.09	42.16	14.44	30.29	68.97	43.92	1.41	21.23	6.65	11.07	48.37	22.89
Meta-Chunking-MSP	6.40	43.01	14.87	31.02	69.36	44.73	79.75	90.92	87.71	8.03	45.83	17.52
RAPTOR	6.61	43.58	15.16	32.66	70.81	46.12	89.24	95.53	93.42	7.87	45.67	17.24
LightRAG	8.22	47.72	17.22	36.89	73.42	50.25	88.69	95.34	93.03	17.04	57.91	29.42
PropRAG	8.83	49.03	18.00	38.71	74.83	51.73	92.07	96.86	95.05	20.31	61.99	32.76
MoC	9.27	49.86	18.59	39.35	75.26	52.29	92.86	97.23	95.51	22.31	64.22	34.74
SAKI-RAG	9.14	50.02	18.27	41.12	76.58	53.70	91.90	99.93	91.96	26.93	82.36	32.70
SPROUTRAG	**13.44**	**57.23**	**23.49**	**49.16**	**81.21**	**60.53**	**100.00**	**100.00**	**100.00**	**38.13**	**89.09**	**42.80**

Table 6: Retrieval performance at depth k=5 across four benchmarks (IE↑, Precision↑, Recall↑). **Bold** = best; underline = second-best. SPROUTRAG rows are shaded.

Method / Unit	Retrieved Evidence	Analysis
Query: *Does the agreement limit the provider's liability, and what exceptions or exclusions apply?*		
MoC	**Retrieved chunk: liability cap.** *Provider's aggregate liability under this Agreement shall not exceed the fees paid by Client during the twelve (12) months preceding the event giving rise to the claim.*	MoC identifies a locally coherent chunk around the monetary cap, but the retrieved unit is too narrow for the query. It answers *how much* liability is capped, but misses the excluded damages, the scope across legal theories, and the exceptions.
SAKI-RAG	**Retrieved pairwise expansion: damage exclusion + liability cap.** 1. *In no event shall Provider be liable for any indirect, incidental, special, consequential, exemplary, or punitive damages.* 2. *Provider's aggregate liability shall not exceed the fees paid during the prior twelve months.*	SAKI-RAG improves over a single chunk by linking two related sentences. However, the evidence remains pairwise, so it captures the main limitation but misses the later sentence listing exceptions such as indemnification, confidentiality breach, gross negligence, and willful misconduct.
Leaf s_1	*In no event shall Provider be liable for any indirect, incidental, special, consequential, exemplary, or punitive damages arising out of or relating to this Agreement.*	Identifies excluded damages, but does not provide the monetary cap or exceptions.
Leaf s_2	*Provider's aggregate liability under this Agreement shall not exceed the fees paid by Client during the twelve (12) months preceding the event giving rise to the claim.*	Provides the liability cap, but not the scope or carve-outs.
Leaf s_3	*The foregoing limitation shall apply regardless of the form of action, whether in contract, tort, strict liability, or otherwise.*	Clarifies that the limitation applies across legal theories.
Leaf s_4	*The limitations in this Section shall not apply to Provider's indemnification obligations, breach of confidentiality, gross negligence, or willful misconduct.*	Provides the exceptions required for a complete legal answer.
SPROUTRAG internal node $v_{1:4}$	**Retrieved clause-level node containing s_1–s_4:** 1. excluded damages, 2. aggregate liability cap, 3. scope across legal theories, 4. exceptions and carve-outs.	**SPROUTRAG retrieves the middle node containing more than two sentences.** This gives the generator the full limitation-of-liability clause, enabling a complete answer that includes both the limitation and the exceptions.

Table 7: Qualitative comparison on a limitation-of-liability query. MoC retrieves a locally coherent but incomplete chunk, and SAKI-RAG retrieves a related sentence pair that captures the main limitation but misses the exception sentence. SPROUTRAG retrieves an internal clause-level node containing four sentences, allowing the answer to include excluded damages, the liability cap, legal-theory scope, and carve-outs.