VecTree-RAG: An Agentic Retrieval-Augmented Generation Framework Combining Vector and Tree Retrieval for Efficiency and Accuracy

Xinyan Zhong[1], Yuwei Shi[1], Yuqi Wei[1], Chen Shen[4], Tianhang Zhou[3], and Zhenghao Wu*[1,2]

[1]*Department of Chemistry and Materials Science, Xi'an Jiaotong-Liverpool University, Suzhou 215123, Jiangsu, P. R. China*
[2]*Department of Chemistry, University of Liverpool, Liverpool L69 7ZD, United Kingdom*
[3]*College of Carbon Neutrality Future Technology, State Key Laboratory of Heavy Oil Processing, China University of Petroleum (Beijing), Beijing 102249, China*
[4]*Suzhou Lab, Suzhou, P. R. China*

arXiv:2607.23006v1 [cs.IR] 25 Jul 2026

Abstract

Scientific question answering requires a retrieval system to solve two distinct problems: identifying which papers are relevant and locating the supporting evidence within those papers. Conventional retrieval-augmented generation typically addresses both through similarity search over fixed-length passages, flattening document structure and separating scientific claims from their methodological and argumentative context. We present VecTree-RAG, an agentic framework that assigns these tasks to complementary retrieval mechanisms. Vector search ranks compact document and section representations across the corpus, whereas reasoning-guided traversal of source-verified section trees localizes evidence within shortlisted papers. Full text is retained in a page store and exposed progressively only after structural localization. We evaluate VecTree-RAG on 300 QASPER questions, an open-access subset of 54 LitQA2 questions, and 49 multi-document MOSAIC questions. Compared with Dense RAG, reranked Dense RAG, RAPTOR, and Search-o1, VecTree-RAG obtained the highest observed answer score on all three benchmarks, reaching 0.800 LLM-judge correctness on QASPER, 0.925 accuracy on LitQA2, and a 0.547 composite score on MOSAIC. On QASPER, its evidence-page precision was 0.274, compared with 0.046–0.071 for the baselines. LitQA2 ablations further showed that the complete vector–tree architecture required fewer inference tokens than variants without tree navigation or corpus-level vector routing. These results indicate that vector retrieval narrows the corpus-level search space and tree navigation concentrates reading on structurally relevant evidence. Although multi-turn inference remains more expensive than single-call retrieval, VecTree-RAG provides a structure-aware and traceable architecture for scientific literature question answering.

Keywords: retrieval-augmented generation; agentic retrieval; document structure; source-verified indexing; progressive disclosure; scientific question answering.

1 Introduction

Retrieval-augmented generation (RAG) commonly represents a document collection as fixed-length passages stored in a vector index (Lewis et al., 2020; Gao et al., 2023b). This representation is

*Email: zhenghao.wu@xjtlu.edu.cn

convenient, but it discards the section hierarchy that organizes a scientific paper. Consequently, retrieved passages can be separated from the methods, figures, or qualifications needed to interpret them. The number of indexed vectors also grows with the number of passages, even when most queries require only a small region of each paper.

Retrieving more passages does not necessarily resolve this problem. Expanding the retrieved set increases redundancy and can obscure useful evidence (Liu et al., 2024), while flat passages provide no representation of how results, methods, and qualifications relate within a paper. These limitations motivate a retrieval architecture that exposes document structure before full text.

VecTree-RAG is an agentic RAG framework that combines two complementary retrieval mechanisms. *Vector retrieval* ranks compact document and section representations across the corpus using semantic and lexical signals, whereas *tree retrieval* navigates the recovered section hierarchy within shortlisted documents to localize pages for closer reading. This *locate-then-read* design treats a paper's authored structure as a retrieval interface rather than flattening it into independent passages. Here, efficiency denotes a compact structural index and selective access to source text; cumulative multi-turn inference cost is evaluated separately in Appendix C.

This paper makes three contributions.

- We introduce VecTree-RAG, an agentic RAG framework built on the insight that corpus-level paper discovery and within-paper evidence localization require different retrieval mechanisms. The framework assigns semantic ranking across documents to vector retrieval and structure-aware navigation within documents to tree retrieval, coordinated by an agent through a locate-then-read workflow (Section 3).
- We develop a source-verified hierarchical index and progressive-disclosure retrieval interface that links document and section representations to their original pages, enabling compact indexing and traceable evidence localization without a passage-level vector store. The interface retains page-resolved source evidence to support sentence-level citations in generated answers (Sections 3.2 and 3.3).
- We introduce MOSAIC (Multi-dOcument Scientific Aggregation, Inference, and Comparison), a benchmark dataset for multi-paper scientific synthesis with 49 free-text questions over 124 papers. Each question requires evidence from five to seven papers and tests aggregation, comparison, or multi-hop inference, with reference answers and document–page evidence for evaluating answer quality and evidence localization (Section 4.1).

2 Related Work

Chunk-based retrieval-augmented generation. The standard RAG pipeline couples a dense retriever (Karpukhin et al., 2020; Khattab and Zaharia, 2020) with a generator over fixed-size passages (Lewis et al., 2020). Hybrid retrieval combines dense and lexical signals such as BM25 (Robertson and Zaragoza, 2009), often through reciprocal rank fusion (Cormack et al., 2009). Query transformations such as HyDE can further improve zero-shot retrieval (Gao et al., 2023a). VecTree-RAG applies dense–sparse fusion to document and section representations, but does not embed raw passages. Its index footprint therefore depends on the number of structural entries rather than the number of text chunks.

Hierarchical retrieval and document navigation. RAPTOR (Sarthi et al., 2024) constructs an induced hierarchy by recursively clustering embedded text segments and summarizing the text in each cluster, then retrieves nodes by similarity. MemWalker (Chen et al., 2023) treats long-document

reading as traversal over a hierarchically summarized memory tree. PageIndex is closely related in retrieval philosophy: its initial design navigates pages and sections through LLM reasoning without a passage-level embedding store (Zhang et al., 2025), and its file-system extension applies hierarchical reasoning to corpus-level routing (PageIndex Team, 2026). VecTree-RAG differs in how it divides retrieval across levels. Vector retrieval performs cross-document paper discovery, whereas tree retrieval navigates the recovered source-authored section hierarchy within shortlisted papers. The hierarchy is grounded in source pages and exposed through a progressive-disclosure interface that retains page-resolved evidence.

Agentic and iterative retrieval. Interleaving retrieval with reasoning can improve multi-hop question answering (Trivedi et al., 2023; Press et al., 2023). ReAct-style tool use (Yao et al., 2023) generalizes this strategy to broader action spaces. Self-RAG (Asai et al., 2024) learns when to retrieve and critique generated responses. PaperQA (Lála et al., 2023) gathers scientific evidence through chunk-level search, whereas Search-o1 (Li et al., 2025) performs iterative search over flat passages. VecTree-RAG instead exposes search results, section summaries, and page content as progressively larger payloads.

Scientific document QA. In scientific-document question answering, QASPER (Dasigi et al., 2021) established information-seeking QA over full research papers and identified evidence selection as a central bottleneck. SPECTER (Cohan et al., 2020) and SPECTER2 (Singh et al., 2023) provide document representations trained using citation structure. VecTree-RAG uses these representations for cross-document retrieval over titles, descriptions, and section summaries rather than raw passages.

3 Method

VecTree-RAG follows a locate-then-read design (Fig. 1). The index preserves the section structure recovered from each paper. Vector similarity ranks compact cross-document representations, while within-document localization uses titles and summaries. The progressive-disclosure interface exposes search results, section structures, and page content in increasing payload sizes.

VecTree-RAG separates document ingestion from query-time retrieval. During ingestion, each PDF is converted into a source-verified section structure; compact document and section representations are indexed, while page text and figures are retained in separate source stores (Section 3.2). At query time, the evaluated agent uses seven tools across three progressive-disclosure layers (Sections 3.3–3.4).

3.1 Preliminaries: Agentic Search

The query-time controller is a ReAct-style agent (Yao et al., 2023) that interleaves reasoning and tool use. Given a question q, the interaction runs for at most L rounds. At round ℓ the state s_ℓ is the message history: the system prompt, the query, and the trajectory of prior tool calls and observations. The policy samples an action

$$a_\ell \sim \pi_\theta(\cdot \mid s_\ell), \qquad a_\ell \in \{\textsc{Final}\} \cup \mathcal{A}, \tag{1}$$

where an action is either a final answer or a tool invocation $a_\ell = (\mathit{name}, \mathbf{x})$ with arguments $\mathbf{x}$. Executing a tool yields an observation o_ℓ, and the state grows by concatenation,

$$s_{\ell+1} \leftarrow s_\ell \oplus (a_\ell, o_\ell). \tag{2}$$

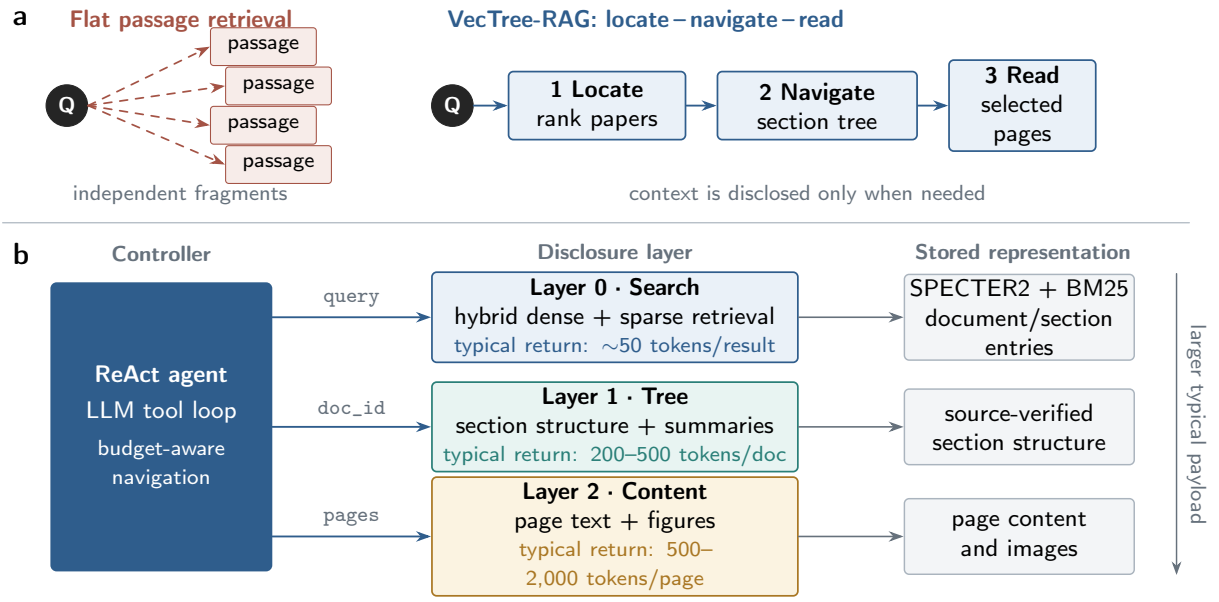

Figure 1. Architecture and progressive disclosure in VecTree-RAG. a, Flat RAG returns independent passages. VecTree-RAG instead ranks papers, navigates a source-verified section structure, and reads selected pages. **b,** The corpus agent moves from compact search results to section summaries and then page text or figures; the right column shows the corresponding stores. Token ranges denote typical serialized tool returns (design estimates), not total model context or fixed costs. Only document and section representations are embedded.

In VecTree-RAG the action set $\mathcal{A}$ comprises seven tools spanning three disclosure layers. Layer 0 provides `search_documents` and `get_paper_metadata`; Layer 1 provides `get_document_tree` and `search_within_paper`; and Layer 2 provides `get_page_content`, `list_images`, and `get_page_image`. Sections 3.3–3.4 describe their roles in the locate-then-read workflow.

3.2 Document Structure Modeling and Verified Indexing

We model each document d as its native section hierarchy, a tree T_d whose nodes are the document's own sections. A node is

$$\nu = \big(id(\nu),\ title(\nu),\ summary(\nu),\ [\,p^{\uparrow}(\nu), p^{\downarrow}(\nu)\,]\big), \tag{3}$$

where $[\,p^{\uparrow}, p^{\downarrow}\,]$ is the inclusive page range that grounds the node in the parsed source. Indexing produces three artifacts: the tree T_d, a page-contents file, and a directory of extracted figures. The tree contains hierarchy, summaries, and metadata, but no body text. The LLM-assisted stages are paired with deterministic checks where source-grounded validation is available.

Parsing. PDFs are converted to per-page markdown with the Marker parser (Paruchuri, 2024), which preserves page boundaries and extracts embedded images with page numbers and bounding boxes. Page indices are the coordinate system for everything downstream: every tree node ultimately resolves to an inclusive page range.

Structure extraction with verification. An LLM reads overlapping page groups, each bounded at 20,000 tokens with one page of overlap. It emits entries $\{(s_i, t_i, p_i)\}$, where s_i is a dotted index, t_i is a title, and p_i is a start page. Each title must match the claimed page within a $\pm$1-page

tolerance using a case-insensitive substring rule. We call the fraction of entries passing this rule the *section-title verification rate*. This measure checks title occurrence and approximate location; it does not validate summary faithfulness or the complete hierarchy. Failed entries are re-extracted from focused candidate pages for at most three repair rounds. A second LLM pass determines whether a title begins its start page, which resolves pages shared by adjacent sections. Finally, a deterministic scan inserts an abstract node when needed.

Tree construction and bounded splitting. The verified flat list is folded into a tree using the dotted indices. Each page range is computed from the node's start page and the next verified heading in document order, before the hierarchy is constructed. Nodes exceeding *both* 20,000 tokens and 10 pages are split by recursively rerunning structure extraction on the node text. This rule bounds nodes that exceed both thresholds, but it does not guarantee a universal maximum on either dimension. Nodes receive stable zero-padded identifiers in depth-first order.

Enrichment and metadata. Each node is summarized from its section text. Parent summaries use preamble text, whereas leaf summaries use the complete body. Sections shorter than 200 tokens retain their source text instead of an LLM summary.

At document level, the pipeline produces a one-sentence description, domain keywords, and bibliographic metadata. Metadata processing begins with regular-expression extraction of DOI, year, title, journal, volume, and pages. An LLM then extracts fields using these candidates as hints. Finally, Crossref verifies and enriches available fields under explicit precedence and format-validation rules. Enrichment calls run concurrently under a semaphore.

The resulting tree contains titles, page ranges, identifiers, and summaries, but no body text. In the current interface, a serialized tree is estimated to return approximately 200–500 tokens for a typical paper. We treat this range as a design estimate because the evaluated corpora do not include a reported distribution of serialized tree sizes.

3.3 Progressive-Disclosure Interface

The query-time controller is a ReAct-style tool loop with an approximately 400-token system prompt. The prompt asks the agent to search before opening documents, inspect tree summaries before requesting pages, and cite page-level evidence. The evaluated loop permits up to 30 iterations. Every model-issued tool call is counted, including repeated, malformed, or failed calls, and a synthesis reminder is appended from the tenth call onward. Page requests are capped at ten pages per call. Image requests return figures as vision inputs when the text layer does not capture their content.

Table 1 summarizes the typical payload returned by each layer. These values describe serialized tool outputs rather than total API-billed tokens, which also include prompts, history, and generated text.

Algorithm 1 states the loop.

3.4 Tools and Synergy

Inter-document retrieval (Layer 0). The first disclosure layer identifies candidate papers. Each *document entry* contains the title, description, authors, journal, and top-level section titles. Each top-level section contributes a separate *section entry* containing its title and summary. Both entry types are short, abstract-like texts.

Dense scores are cosine similarities between the L2-normalized SPECTER2 embedding (768-d, 512-token input cap) of the query and each entry. Sparse scores come from BM25 over lowercased

Layer	Tool	Returns	Payload estimate (tokens)
0	`search_documents`	metadata + matching sections	∼50 / result
	`get_paper_metadata`	metadata + top sections	∼50 / doc
1	`get_document_tree`	section structure + summaries	200–500 / doc
	`search_within_paper`	matching section titles	query-dependent
	`get_page_content`	full-page markdown (≤10 pages/call)	500–2,000 / page
	`list_images`	available figure filenames	file-count-dependent
	`get_page_image`	base64 figure (vision input)	model-dependent

Table 1. Progressive-disclosure tool interface. Payload estimates refer to serialized tool returns and exclude the accumulated prompt, conversation history, and model output.

alphanumeric tokens of the same entries. Per document, the best-scoring entry of each type is taken, so a paper can be surfaced either because it is globally about the query or because one section is. Candidates must pass a pre-fusion relevance gate (dense similarity ≥ 0.35 or BM25 score ≥ 1.0) that discards documents which neither retriever considers relevant, rather than letting rank fusion promote them. Surviving candidates are fused by reciprocal rank fusion with the standard constant $k = 60$:

$$\mathrm{RRF}(d) \;=\; \frac{1}{k + r_{\mathrm{dense}}(d)} \;+\; \frac{1}{k + r_{\mathrm{sparse}}(d)}, \tag{4}$$

where $r_\cdot(d)$ is the document's rank under each retriever. A post-fusion gate adapts to corpus size. Above 50 documents, it removes results below 40% of the leading RRF score. For smaller corpora, it requires retriever agreement or a strong individual match. The strong-match thresholds are 0.55 for dense similarity and 4.0 for BM25. These gates filter marginal results before they are presented to the agent.

Each returned result carries only metadata (identifier, title, authors, venue, year, page count, one-sentence description, and the matching section titles), at roughly 50 tokens. The matching-sections field bridges to the next layer: it tells the agent which paper to open and where in it to look first.

Within-document navigation and standalone verified tree search (Layer 1). The evaluated corpus agent loads titles, summaries, and page ranges through `get_document_tree`. It then selects pages directly within its tool loop. This navigation step does not invoke a separate deterministic validation procedure.

The standalone single-document mode uses an additional verified tree-search procedure. An LLM returns candidate node identifiers, which are checked for existence, valid page ranges, and lexical overlap with the query. The fraction of passing nodes defines a rule-based confidence score. Scores below 0.5 trigger at most two retries with an error report. If confidence remains low, a lexical fallback ranks nodes by stopword-filtered overlap and returns the top five. This validation checks output conformity, not whether the selected sections contain the correct answer.

Complexity and index footprint. For a corpus of N documents with an average of S indexed section entries per document, the vector index contains $N(1+S)$ embeddings. A chunk store instead contains a number of vectors proportional to total corpus tokens divided by chunk size. We do not report a corpus-matched storage benchmark, so we restrict the claim to this difference in index construction. Query-time cost is dominated by repeated LLM calls and is reported in Appendix C.

Algorithm 1 VecTree-RAG query-time agent loop (locate-then-read).

Require: Query q; corpus index $\mathcal{I}$ (SPECTER2+BM25 over doc/section entries); metadata registry $\mathcal{M}$; source-verified structures $\{T_d\}$; page and image stores $\mathcal{P}$; iteration cap L

$s_1 \leftarrow [\textsc{System},\ \textsc{User}\colon q]$
$c \leftarrow 0$ ▷ number of model-issued tool calls
for $\ell = 1$ **to** L **do**
  $a_\ell \sim \pi_\theta(\cdot \mid s_\ell)$
  if $a_\ell = \textsc{Final}$ **then**
    return answer
  end if
  $c \leftarrow c + 1$
  if $a_\ell.name =$ `search_documents` **then**
    $o_\ell \leftarrow \textsc{HybridRetrieve}(a_\ell.\mathbf{x}, \mathcal{I})$ ▷ Layer 0
  else if $a_\ell.name =$ `get_paper_metadata` **then**
    $o_\ell \leftarrow \textsc{MetadataLookup}(a_\ell.doc, \mathcal{M})$ ▷ Layer 0
  else if $a_\ell.name =$ `get_document_tree` **then**
    $o_\ell \leftarrow T_{a_\ell.doc}$ ▷ Layer 1
  else if $a_\ell.name =$ `search_within_paper` **then**
    $o_\ell \leftarrow \textsc{HybridRetrieve}(a_\ell.\mathbf{x}, \mathcal{I} \mid a_\ell.doc)$ ▷ Layer 1
  else if $a_\ell.name =$ `get_page_content` **then**
    $o_\ell \leftarrow \mathcal{P}[\, a_\ell.doc,\ a_\ell.pages\,]$ ▷ Layer 2 text
  else if $a_\ell.name =$ `list_images` **then**
    $o_\ell \leftarrow \textsc{ImageList}(a_\ell.doc, \mathcal{P})$ ▷ Layer 2 index
  else if $a_\ell.name =$ `get_page_image` **then**
    $o_\ell \leftarrow \textsc{ImageRead}(a_\ell.\mathbf{x}, \mathcal{P})$ ▷ Layer 2 vision
  end if
  if $c \geq 10$ **then** append a synthesize-now note
  $s_{\ell+1} \leftarrow s_\ell \oplus (a_\ell, o_\ell)$
end for
return most recent nonempty assistant text; otherwise raise an evaluation error

Indexing requires structure extraction and summarization; Appendix D reports its measured wall time.

Composition. Layer 0 ranks papers and returns the titles of their best-matching sections. Layer 1 exposes the source-verified section structure for direct agent navigation. Layer 2 provides page text and images for close reading. The ablations in Section 4.6 test the vector and tree components within the evaluated corpus agent.

4 Experiments

We evaluate VecTree-RAG in three regimes. QASPER tests question answering within one known paper. LitQA2 tests corpus-level retrieval when one paper contains the answer. MOSAIC tests synthesis across at least five papers. The main evaluation reports answer quality and evidence localization at the granularity available for each benchmark. Cumulative API-billed tokens and wall time are reported in Appendix C.

4.1 Benchmark Details

QASPER (single-document). We use a stratified sample of 300 questions drawn once with a fixed sampling seed from the QASPER test split (Dasigi et al., 2021). The complete ingested split contains 1,451 questions across 416 papers. QASPER's supplied section structure is represented with one synthetic page per paragraph. Gold paragraph evidence therefore aligns with the page identifiers used by the agent.

LitQA2 (corpus-level single-source QA). We use an open-access subset of LitQA2 from LAB-Bench (Laurent et al., 2024). Gold DOIs were resolved through OpenAlex, Unpaywall, and Europe PMC, yielding 62 accessible PDFs. Marker parsing and indexing succeeded for 61 papers. After excluding questions without an indexed gold paper, 54 questions remained in a closed 61-paper corpus. This accessibility filter can introduce selection bias relative to the full benchmark.

MOSAIC (multi-document synthesis). MOSAIC contains 49 free-text questions over 124 materials-science and computational-science papers. Each question requires evidence from five to seven papers. The set includes 24 inference, 13 aggregation, and 12 comparison questions. Each item has a free-text reference answer, an explicit reasoning chain, and gold (document, page) evidence pairs. Every system was evaluated on all 49 questions across three seeds.

MOSAIC construction and quality control. Candidate paper groups were formed as dynamic topical neighborhoods using SPECTER2 similarity over titles and document descriptions, combined with keyword overlap. DeepSeek-V4-Pro drafted one question, reference answer, reasoning chain, and evidence set at a time from the selected source pages at temperature 0.35. Aggregation questions collect directly stated facts across papers; comparison questions contrast a shared axis; and inference questions require a conclusion not stated by any single paper. Automated checks at temperature 0.0 tested source grounding, answerability, closed-book leakage, multi-paper coverage, and duplication. Inference questions additionally passed single-document sufficiency and chunk-retrieval checks. Failed grounding or coverage checks triggered a bounded repair loop. The released set was constructed and screened automatically without independent human validation or adjudication.

4.2 Baselines, Systems, and Protocol

Systems. All reported experiments used repository snapshot `a57c300` and the seven-tool interface described in Section 3.4. We compare the full VecTree-RAG agent, with a 30-iteration cap, against four baselines used consistently across the three benchmarks. On QASPER and LitQA2, Dense RAG retrieved the top ten 800-token passages with 400-token overlap. On MOSAIC, it used an expanded-context setting with $k = 232$ to test whether a large flat context could recover evidence across many papers. This setting is a retrieval-volume stress test rather than an exact match of realized API cost. Dense RAG with reranking rescored 30 dense candidates with a BGE cross-encoder (Xiao et al., 2024) and retained ten. RAPTOR (Sarthi et al., 2024) retrieved from a recursive summary tree. Search-o1 (Li et al., 2025) used a ten-round flat-passage search loop. QASPER retrieval systems received the gold paper, whereas LitQA2 systems searched the complete corpus.

Generation, prompts, and grading. All answer-generation systems used DeepSeek-V4-Flash through OpenRouter at temperature 1.0. Each system was evaluated with three generation seeds, and reported metrics are means over the three seed-level results. DeepSeek-V4-Pro at temperature 0.0 judged every system under a shared rubric. On QASPER, it assessed correctness against the

System	QASPER correctness ↑	LitQA2 accuracy ↑	MOSAIC composite ↑
Dense RAG	0.750	0.870	0.503
Dense RAG w/ Reranker	0.757	0.889	0.334
RAPTOR	0.690	0.815	0.329
Search-o1	0.757	0.759	0.489
VecTree-RAG (agent)	**0.800**	**0.925**	**0.547**

Table 2. Benchmark-specific answer scores. Values are means over three seeds on fixed sets of 300 QASPER, 54 LitQA2, and 49 MOSAIC questions. QASPER and LitQA2 use DeepSeek-V4-Pro judgments at temperature 0.0, whereas MOSAIC uses 0.75 factuality-F1 + 0.25 scaled SPECTER cosine. The MOSAIC Dense RAG entry uses the expanded-context setting with $k = 232$; QASPER and LitQA2 use $k = 10$. The metrics are not commensurate and are therefore not averaged across benchmarks.

official answers; this metric is distinct from official token-level F1. On LitQA2, it assessed whether the response selected the correct option. Available percentile-bootstrap intervals quantify variation across questions.

4.3 Evaluation Metrics

All metrics are computed per question and macro-averaged within each seed; tables report the mean of the three seed-level values. QASPER correctness is the proportion of responses assigned *correct* by the judge. A *partial* judgment receives zero in this strict headline metric. LitQA2 accuracy is the proportion of responses that select the gold option.

Evidence precision and recall are computed from sets, so repeated reads of the same evidence unit are counted once. QASPER uses paragraph-level synthetic page identifiers and scores against the best-matching annotator evidence set. A retrieved baseline chunk inherits every paragraph identifier that it spans. MOSAIC instead uses exact (document, page) pairs. LitQA2 key passages are mapped to physical pages by source-text matching; only 10 of 54 questions have a nonempty mapped gold-page set. We therefore treat LitQA2 page scores as an exploratory mapping diagnostic rather than a corpus-level retrieval metric.

For MOSAIC, the judge decomposes an answer into atomic statements and compares them with the reference statements to obtain factuality precision, recall, and F1. Coverage is the fraction of reference facts attributed to the response, and faithfulness is the fraction of response statements judged inferable from the retrieved context. Semantic similarity uses `allenai/specter2_base` with a 512-token input cap, the 768-dimensional `[CLS]` representation, and L2 normalization. Cosine similarity is scaled to $s_{\cos} = (\cos + 1)/2$, and the reported composite is

$$\text{Composite} = 0.75\,\text{F1}_{\text{factuality}} + 0.25\,s_{\cos}. \tag{5}$$

4.4 Main Results

Single-document QASPER. VecTree-RAG obtained a mean LLM-judge correctness rate of 0.800 across three seeds on the 300-question sample, higher than all four evaluated baselines (Table 2). The corresponding scores were 0.750 for Dense RAG, 0.757 for Dense RAG with reranking, 0.690 for RAPTOR, and 0.757 for Search-o1. VecTree-RAG also reached evidence-page precision of 0.274, compared with 0.046–0.071 for these baselines, while retaining evidence recall of 0.737 (Table 3). Thus, its accuracy advantage was accompanied by a more concentrated set of source evidence.

System	Evidence recall ↑	Evidence precision ↑
Dense RAG	0.884	0.048
Dense RAG w/ Reranker	0.894	0.048
RAPTOR	**0.899**	0.046
Search-o1	0.817	0.071
VecTree-RAG (agent)	0.737	**0.274**

Table 3. QASPER evidence localization. Values are means over three seeds. Recall and precision compare deduplicated evidence sets with gold paragraph identifiers. Retrieved baseline chunks inherit all synthetic identifiers that they span. The values therefore measure paragraph-level evidence selection rather than localization to physical PDF pages.

System	Composite ↑	Coverage ↑	Faith ↑	Ev.R ↑	Ev.P ↑
Dense RAG (expanded, $k = 232$)	0.503	0.294	0.033	**0.661**	0.019
Dense RAG w/ Reranker	0.334	0.114	**0.174**	0.153	0.076
RAPTOR	0.329	0.073	0.097	0.404	0.050
Search-o1	0.489	0.318	0.134	0.411	0.086
VecTree-RAG (agent)	**0.547**	**0.532**	0.170	0.586	**0.221**

Table 4. MOSAIC multi-document synthesis. Values are means over three seeds on all 49 questions. The composite is 0.75 factuality-F1 + 0.25 scaled SPECTER cosine. Ev.R and Ev.P denote document-and-page-aware evidence recall and precision. Dense RAG uses an expanded-context setting with top-$k = 232$; this is not an exact match of realized API cost.

Corpus-level LitQA2. VecTree-RAG achieved a mean accuracy of 0.925 across three seeds (95% question-bootstrap CI, 0.85–0.98), higher than all four evaluated baselines. The corresponding scores were 0.870 for Dense RAG, 0.889 for Dense RAG with reranking, 0.815 for RAPTOR, and 0.759 for Search-o1 (Table 2). The result shows that the vector–tree agent can preserve high answer accuracy while locating the relevant source within a 61-paper corpus. Query-time computation is reported in Appendix C.

MOSAIC. The MOSAIC composite combines factuality-F1 with scaled SPECTER similarity; coverage and faithfulness use the same judge family. Evidence metrics require the correct document and page. VecTree-RAG achieved a composite score of 0.547 and coverage of 0.532, both higher than the same four evaluated baselines (Table 4). It also obtained the highest observed evidence precision, 0.221, while maintaining evidence recall of 0.586. Expanded-context Dense RAG reached higher evidence recall, 0.661, but with precision of 0.019. Thus, in the multi-document setting, VecTree-RAG combined strong synthesis quality with a substantially more concentrated set of document–page evidence.

4.5 Source-Traceable Evidence Localization

VecTree-RAG read a smaller fraction of irrelevant QASPER pages, as reflected by evidence precision of 0.274 compared with 0.046–0.071 for the four retrieval baselines. The corresponding recall of 0.737 was lower than the baseline range of 0.817–0.899. The method therefore improved the concentration of relevant evidence rather than complete evidence coverage. Because each selected item resolves to a source page or QASPER paragraph identifier, the resulting evidence set can be inspected directly

against the original document.

Verifiability operates at two bounded levels. During indexing, deterministic checks confirm section-title occurrence and approximate source location. During evaluation, evidence metrics compare deduplicated retrieved units with benchmark-native gold evidence. These checks make the retrieval basis traceable, but they do not establish the faithfulness of generated summaries or the factual correctness of every answer. The harness preserves the set of read pages but not the ordered tool trajectory, so adherence to the intended layer order cannot yet be quantified.

4.6 Component Ablations

On LitQA2, we remove the tree layer, vector layer, or relevance gates from the corpus agent. The standalone verify–retry–fallback procedure is characterized separately in Appendix D. Progressive disclosure itself was not isolated as an ablation.

Variant (agent)	**MCQ accuracy** ↑ [95% CI]
Full system	**0.925** [0.85, 0.98]
− tree layer (search → pages)	0.904 [0.83, 0.98]
− vector layer (tree only, all docs)	0.811 [0.70, 0.91]
− relevance gates (raw RRF)	0.906 [0.81, 0.98]

Table 5. LitQA2 component ablations. Accuracy values are means over three seeds; brackets show question-bootstrap 95% CIs. The table reports descriptive component comparisons without inferential p-values. Limited page-mapping diagnostics are reported in Appendix B.

5 Discussion and Limitations

Across three scientific-document settings, VecTree-RAG achieved the highest observed answer score relative to Dense RAG, Dense RAG with reranking, RAPTOR, and Search-o1. It reached 0.800 LLM-judge correctness on QASPER, 0.925 accuracy on LitQA2, and a 0.547 MOSAIC composite score. The advantage was accompanied by more selective evidence localization: QASPER evidence precision was 0.274, compared with 0.046–0.071 for the four baselines, while MOSAIC evidence precision reached 0.221. These results indicate that combining cross-document vector retrieval with within-document tree retrieval can improve observed answer quality while concentrating retrieval on source-relevant pages.

Reasoning-based hierarchical systems such as MemWalker and PageIndex are attractive for within-document navigation because an LLM can inspect textual section representations and decide which branch to read (Chen et al., 2023; Zhang et al., 2025). PageIndex has also described a query-dependent file-level hierarchy that extends tree search to corpus-level routing (PageIndex Team, 2026). This development highlights an architectural choice: LLM traversal can exploit an informative hierarchy, but broad or weakly labeled branches may require larger routing views or additional reasoning steps. VecTree-RAG takes a complementary design point, using vector retrieval to rank compact document and section representations across the corpus and reserving tree reasoning for shortlisted papers. This specialization limits LLM-based structural navigation to a small candidate set while retaining page-resolved source text for sentence-level citations.

The LitQA2 ablation provides internal evidence for this architectural specialization. When vector routing was removed and the agent navigated trees across all 61 papers, mean cumulative API-billed tokens increased from 127,872 to 490,482 per completed query, while observed accuracy decreased

from 0.925 to 0.811. Removing tree navigation also increased token use to 194,821. These results suggest that vector routing is particularly important for reducing the corpus-level reasoning space, whereas tree navigation provides a more economical interface to source text after document selection. Removing the relevance gates reduced token use to 104,215 and produced a similar observed accuracy of 0.906, so the gates should be viewed as routing heuristics rather than an independently established efficiency gain. At the index level, VecTree-RAG stores embeddings for document and section representations rather than fixed-length passages. Its vector index therefore scales with structural entries rather than corpus chunk count.

The efficiency claim is therefore scoped to index representation and navigation within the agentic design; it does not imply uniformly lower end-to-end inference cost. The multi-turn agent used more query-time tokens than Dense RAG, Dense RAG with reranking, and RAPTOR; relative to Search-o1, its cost was higher on QASPER but lower on LitQA2 and MOSAIC. Detailed measurements are reported in Appendix C. A fuller efficiency comparison should jointly account for embedding counts, serialized index size, indexing cost, retrieval latency, and cached-prefix tokens. Smaller navigation models, prefix caching, and context compaction may reduce query-time computation while preserving evidence traceability.

VecTree-RAG also has method-level boundaries. It assumes informative section structure and may transfer poorly to unstructured documents. Navigation depends on generated summaries whose faithfulness is not currently verified, and page-level grounding succeeds only when the agent chooses the relevant section. Page-resolved sources can support sentence-level citations, but the present evaluation measures evidence localization only at paragraph or page granularity. MOSAIC was generated and screened automatically without independent human validation, and its construction and scoring use the same model family. Future work should therefore add expert adjudication, validate generated section summaries, record ordered tool trajectories, and test the method on larger and less regularly structured corpora.

6 Conclusion

VecTree-RAG assigns corpus-level paper discovery to vector retrieval and within-paper evidence localization to tree retrieval. Against Dense RAG, Dense RAG with reranking, RAPTOR, and Search-o1, it achieved the highest observed answer score on QASPER (0.800), LitQA2 (0.925), and MOSAIC (0.547). It also produced substantially higher QASPER evidence precision (0.274 versus 0.046–0.071) and the highest observed MOSAIC evidence precision (0.221). The source-verified hierarchy links compact document and section representations to original pages, providing an inspectable basis for page-resolved citations. These findings support complementary retrieval mechanisms for accurate and traceable scientific question answering.

Code Availability

The source code, evaluation harness, and dataset-construction pipeline are available in the VecTree-RAG repository.

A Dataset and Corpus Statistics

Table 6 reports the question and corpus statistics for the three evaluation regimes. The QASPER study uses a stratified sample of 300 questions drawn once with a fixed sampling seed from the 1,451-question test split (416 papers); the sample touches 215 distinct gold papers and preserves the

official answer-type proportions (extractive 50.3%, abstractive 25.7%, yes/no 14.3%, unanswerable 9.7%). Because QASPER papers are ingested with one synthetic page per paragraph (Section 4.1), the "pages" column for QASPER counts paragraphs rather than physical pages and is therefore not comparable to the two PDF-parsed corpora. LitQA2 and MOSAIC page counts are physical pages recovered by Marker and read directly from each tree's `total_pages` field. The evidence row records benchmark-native annotations: gold paragraphs for QASPER, one supplied key passage for each LitQA2 question, and document-level evidence records with page locations for MOSAIC. Only 10 LitQA2 key passages were matched to physical pages, so its mapped-page statistics are reported separately as an exploratory diagnostic.

	QASPER (single-doc)	**LitQA2** (multi-doc)	**MOSAIC** (≥5-paper)
Question composition			
Benchmark questions	300	54	49
extractive / abstractive	151 / 77	—	—
yes-no / unanswerable	43 / 29	—	—
MCQ	—	54	—
multi-hop inference	—	—	24
aggregate / compare	—	—	13 / 12
Corpus coverage			
Corpus papers	416	61	124
Gold papers (in study)	215	54	124
Publication years	—	2012–2024	2012–2026
Document length			
Paragraph units/paper (mean)	49.6	—	—
Paragraph units/paper (median; range)	44.5; 7–309	—	—
Physical pages/paper (mean)	—	21.8	12.8
Physical pages/paper (median; range)	—	18; 1–57	12; 6–34
Evidence			
Gold evidence records/q (mean)	2.5	1.0	5.1
Gold papers/q (range)	1	1	5–7

Table 6. Benchmark and corpus statistics. Answer-type rows sum to the benchmark-question total. Evidence records differ in granularity: QASPER supplies gold paragraphs, LitQA2 supplies one key passage per question, and MOSAIC supplies multi-paper evidence with physical page locations.

B LitQA2 Evidence-Mapping Diagnostic

LitQA2 supplies one gold document and one key passage per question rather than complete page-level annotations. Exact source-text matching mapped 10 of 54 key passages to physical pages. Table 7 reports the resulting page-set scores for completeness, but these values are not evidence of corpus-level document precision and do not support the main component claims.

Variant (agent)	Mapped-page recall ↑	Mapped-page precision ↑
Full system	0.151	0.076
− tree layer (search → pages)	0.173	0.109
− vector layer (tree only, all docs)	0.245	0.207
− relevance gates (raw RRF)	0.151	0.108

Table 7. Exploratory LitQA2 page-mapping diagnostic. Values are means over three seeds. Gold pages are available for only 10 of 54 questions after key-passage matching, and predictions are evaluated within the gold document. These values should not be compared directly with the QASPER or MOSAIC evidence metrics.

C Query-Time Inference Cost

Table 8 reports the operational cost measurements separately from the main accuracy and evidence-localization results. VecTree-RAG used more query-time computation than Dense RAG, Dense RAG with reranking, and RAPTOR on QASPER and LitQA2. The comparison measures inference only; it does not include a corpus-matched accounting of embedding construction or serialized index size.

	QASPER		LitQA2	MOSAIC
System	Tokens/q ↓	Wall (s) ↓	Tokens/q ↓	Tokens/q ↓
Dense RAG	7,660	11.0	9,696	194,187
Dense RAG w/ Reranker	7,722	20.2	8,660	16,648
RAPTOR	**5,594**	15.3	9,045	18,917
Search-o1	31,261	25.5	141,173	409,748
VecTree-RAG (agent)	41,186	45.9	127,872	279,337

Table 8. Inference cost across benchmarks. Values are averaged over three seeds. Tokens per query are mean cumulative API-billed totals, and wall time is the QASPER mean. The MOSAIC Dense RAG value uses the expanded-context setting with $k = 232$.

The LitQA2 component removals also changed query-time computation (Table 9). Removing the vector layer produced the largest increase because the agent had to inspect trees across the full corpus.

LitQA2 variant (agent)	Tokens/q ↓
Full system	127,872
− tree layer (search → pages)	194,821
− vector layer (tree only, all docs)	490,482
− relevance gates (raw RRF)	104,215

Table 9. Query-time cost of the LitQA2 component ablations. Values are mean cumulative API-billed tokens per query, averaged over three seeds.

D Indexing and Tree-Search Intrinsics

The standalone verify–retry–fallback procedure is not part of the evaluated corpus-agent loop. We therefore report it as an intrinsic measurement. Across 61 indexed LitQA2 papers, the mean section-title verification rate increased from 0.910 to 0.924 after repair. The corresponding medians were 0.957 and 0.958, with 2.1 repair rounds on average. Papers averaged 21.8 pages, and the LLM indexing stages required 254 s per paper on average, excluding Marker parsing.

On a 100-question QASPER sample, the rule-based node-confidence score was 1.0 for every query, and the lexical fallback was never triggered. This result shows that returned identifiers satisfied the validation rules. It does not measure whether the selected nodes contained the correct answer.

E Judge Robustness

DeepSeek-V4-Pro judged outputs generated by DeepSeek-V4-Flash, creating a possible model-family preference. Deterministic checks assess agreement only where answer extraction is mechanically unambiguous (Table 10). These response-level diagnostic subsets were assembled from available evaluation runs and are not a separate re-evaluation of the five-system main comparison. They do not validate judge behavior on longer free-text responses.

Benchmark	Mechanical subset	Agreements	Agreement
QASPER yes/no	387	383	0.990
LitQA2 option selection	470	469	0.998

Table 10. Agreement between the LLM judge and deterministic extraction on mechanically unambiguous outputs. The response-level diagnostic subsets exclude outputs whose final answer cannot be extracted reliably by a fixed rule.

The same judge rubric was applied to every system, which reduces one source of procedural asymmetry. However, systems differ in answer length and format, so symmetric scoring does not eliminate family or style bias. Evidence-page precision and recall are independent of the judge, but the headline answer scores are not. Regrading all outputs with multiple out-of-family judges remains the appropriate robustness test.

F Prompt Templates

This appendix transcribes selected query-time and grading prompts verbatim from the released code. Corpus-dependent fields (document counts, year ranges) and per-question fields are shown as `{braced}` placeholders filled at runtime. The complete prompt set is available in the released evaluation harness.

Agent system prompt. The query-time ReAct agent is driven by the following prompt (`src/agent.py`, `build_system_prompt`). It is deliberately short (≈400 tokens) and encodes the progressive-disclosure discipline.

```
You are a research assistant with access to a corpus of {total_documents} scientific papers ({
↪ year_range}) across {journal_count} journals. You answer questions by searching and reading
↪ these papers.
```

```
## Workflow
Classify the question, then act:
1. Library question (how many papers, what's available, list by author/year/topic):
   Answer from the corpus stats above when sufficient. Use search_documents("*") with
   filters to list or browse specific papers.
2. Research question (about content, methods, findings, comparisons):
   search_documents(query) -> pick relevant papers -> get_document_tree -> read summaries
   -> get_page_content for key sections -> synthesize answer with citations.
   If 0 results, reformulate with synonyms, broader terms, or author names.
3. Follow-up on papers already discussed:
   Use known doc_ids directly. Search again only if you need additional papers.

Some papers have supporting information (SI). Search results may include
related_documents with SI doc_ids -- use get_document_tree on those when needed.

## Token Budget Rules
- Read tree summaries BEFORE requesting pages.
- Never retrieve more than 5 pages without synthesizing first.
- If you have enough information to answer, stop retrieving.
- Use filters (year_min, year_max, journal, author) to narrow searches when appropriate.

## Citation Format
Cite as: (Author et al., Year, doc_id, p.X).

## Multi-Document Synthesis
When the question requires analysis across papers, synthesize with comparative language
and cite each source inline. If the question asks for a list or identification, a concise
answer is appropriate -- do not add unsolicited analysis sections.

## Images
Page text may contain inline image references like ![](filename.jpeg). Use get_page_image
to view figures and tables. Describe what the figure shows and incorporate it.
```

QASPER correctness judge. Free-text answers are graded by the following judge (`eval/metrics/qasper_regrade.py`), under the system message *"You are a strict grader for free-text questions about scientific papers. Respond with JSON only."*

```
Question: {question}

Reference answers (any one is acceptable):
{references}

A system produced the response below. Decide whether the response's FINAL answer
matches the meaning of ANY reference answer.
- "correct": conveys the same answer as a reference (for yes/no, the same polarity;
  if a reference is "Unanswerable", the response must conclude the question cannot be
  answered from the paper).
- "partial": overlaps a reference but misses or distorts a key part.
- "incorrect": contradicts every reference, or answers a different question.

Response:
"""{answer_text}"""

Reply with ONLY a JSON object: {"verdict": "correct|partial|incorrect",
"rationale": "<one short sentence>"}
```

LitQA2 MCQ judge. Multiple-choice answers are graded by the following judge (`eval/metrics/mcq_judge.py`), under the system message *"You are a strict grader for multiple-choice questions about scientific papers. Respond with JSON only."*

```
Question: {question}

Options:
{lettered_options}

The correct option is {gold_letter}. {gold_text}

A system produced the response below. Decide whether the response, as its FINAL answer,
selects the correct option. Merely mentioning or discussing the correct option's content
is NOT enough; it must be the response's conclusion. Use "abstain" only if the response
concludes there is insufficient information to answer.

Response:
"""{answer_text}"""

Reply with ONLY a JSON object: {"verdict": "correct|incorrect|abstain",
"selected": "<the option letter the response chose, or none>",
"rationale": "<one short sentence>"}
```

MOSAIC synthesis judges. MOSAIC answer correctness follows the GraphRAG-Bench protocol (`eval/metrics/judge.py`): an LLM decomposes each answer into atomic statements, classifies each against the gold statements as true positive, false positive, or false negative to yield a factuality F1, and separately judges faithfulness by testing whether each answer statement is directly inferable from the read context. Answer correctness combines factuality F1 with scaled SPECTER embedding cosine as $0.75\,\mathrm{F1} + 0.25\,s_{\cos}$; coverage is the fraction of extracted reference facts attributed to the response. All MOSAIC sub-judgments use DeepSeek-V4-Pro at temperature 0.0, matching the QASPER and LitQA2 judge. The full prompt text for each sub-judge is released with the harness.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations (ICLR)*, 2024.

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

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

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 758–759, 2009.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4599–4610, 2021.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1762–1777, 2023a.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023b.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48, 2020.

Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G. Rodriques, and Andrew D. White. PaperQA: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*, 2023.

Jon M. Laurent, Joseph D. Janizek, Michael Ruzo, Michaela M. Hinks, Michael J. Hammerling, Siddharth Narayanan, Manvitha Ponnapati, Andrew D. White, and Samuel G. Rodriques. Lab-bench: Measuring capabilities of language models for biology research. *arXiv preprint arXiv:2407.10362*, 2024.

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. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474, 2020.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*, 2025.

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

PageIndex Team. PageIndex File System: Massive-scale document search. PageIndex Blog, May 2026. `https://pageindex.ai/blog/pageindex-filesystem`.

Vikas Paruchuri. Marker: Convert PDF to markdown quickly with high accuracy. `https://github.com/VikParuchuri/marker`, 2024.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *Findings of the Association for Computational Linguistics: EMNLP*, pages 5687–5711, 2023.

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

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

Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. SciRepEval: A multi-format benchmark for scientific document representations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5548–5566, 2023.

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

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general chinese embeddings. *arXiv preprint arXiv:2309.07597*, 2024. BGE reranker (bge-reranker-v2-m3).

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Mingtian Zhang, Yu Tang, and PageIndex Team. PageIndex: Next-generation vectorless, reasoning-based RAG. PageIndex Blog, 2025. `https://pageindex.ai/blog/pageindex-intro`.