Research and Science TODAY
www.rstjournal.com

OPEN ACCESS

Section: *Applied Data Science and Artificial Intelligence*
Issue: *1/2026*
Article number: *21.2026*

DOI
https://doi.org/10.38173/RST.2026.1.21

Summary RAG: A Multi-Format Document Retrieval System with Document-Level Summarization

Daniel HERNANDEZ DE LEON[1]
Oleksandr KONDRATIUK[2]

[1] Data Scientist, akirolabs GmbH, Germany, daniel.deleon@akirolabs.com
[2] CTO, akirolabs GmbH, Germany, olek@akirolabs.com

Received: *8 April 2026*
Revised: *18 May 2026*
Accepted: *18* May 2026
Available online: 28 May 2026

Suggested citation
D. Hernandez de Leon; O. Kondratiuk. "Summary RAG: A Multi-Format Document Retrieval System with Document-Level Summarization", Research and Science Today, vol. 2026, no. 1, art. no. 21.2026, pp. 1–21, 2026, doi: 10.38173/RST.2026.1.22.

Abstract
Retrieval-augmented generation (RAG) systems fragment documents into fixed-size chunks, losing context and poorly representing structured content such as tables, multi-column layouts, and hierarchical information. We propose Summary RAG, an architecture combining document-level summarization with selective full-content access. The system creates semantic summaries for document-level retrieval while preserving original content for detailed answer generation. Evaluation across 180,148 documents and 673 queries on three datasets from the Benchmarking IR (BEIR) suite shows directional improvements on two of three datasets: +16.5% Normalized Discounted Cumulative Gain (NDCG@100) on NFCorpus and +12.2% on TREC-COVID, with a -7.9% NDCG@100 degradation on the fact-dense SciFact dataset. On TREC-COVID we observe a 3.1× speedup in the vector-embedding/indexing phase only (after summarization is complete). A small-scale multi-format pilot covering PDF, DOCX, Markdown, tabular, and image documents (125 documents, 21 queries; LLM-as-judge scoring with gpt-4o) shows +81.3% answer relevancy and +27.8% faithfulness over Naive RAG; given the pilot scale and judging methodology, these results should be interpreted as exploratory rather than conclusive. Total indexing time, including LLM summarization, is approximately 7× slower than Naive RAG, while the retrieval-index vector count is reduced by up to 99.3% on the 125-document pilot (≈50% on TREC-COVID). The approach achieves O(n) vector-indexing complexity versus O(n × c) for chunk-based methods, where c is the average number of chunks per document; summarization cost is also O(n). The scalability advantage applies to the retrieval-index size, not to total system storage, which additionally retains the full document content for generation.

Keywords: *RAG, LLM, Semantic-Search*

INTRODUCTION

Enterprise document processing requires handling diverse content formats, each with specialized characteristics for effective information retrieval. Retrieval-Augmented Generation (RAG) systems apply uniform chunking strategies across heterogeneous document collections, creating limitations that affect performance in real-world deployment scenarios.

Current RAG implementations fragment documents into fixed-size segments without considering document structure, format-specific characteristics, or hierarchical information organization. This approach introduces three key problems: context fragmentation where relationships between document sections are lost, scale limitations as document collections grow and chunk-based retrieval becomes noisy, and query-context mismatch where user queries requiring document-level understanding cannot be satisfied by fragmented chunk retrieval.

Lewis et al. [1] introduced the foundational RAG architecture, combining parametric language models with non-parametric retrieval mechanisms to address knowledge limitations in pre-trained models, establishing the retrieve-then-generate paradigm. Subsequent research expanded RAG capabilities: including Fusion-in-Decoder (FiD) [2], improved multi-passage reading comprehension by processing retrieved passages jointly. Independently, Borgeaud et al. [3] introduced RETRO, scaling retrieval to trillions of tokens through chunked cross-attention. REALM [4] introduced end-to-end pre-training of retrieval components.

This investigation addresses three research questions: Does document-level summarization improve retrieval quality compared to chunk-based approaches? How does Summary RAG perform across different document formats and corpus scales? What are the computational tradeoffs between summarization overhead and retrieval efficiency?

We introduce Summary RAG, an architecture that combines document-level summarization with selective full-content access. Our approach provides language models with both contextual understanding through document summaries and access to granular details when needed, maintaining document coherence while enabling precise information extraction.

RELATED WORK

Traditional RAG implementations rely on document chunking strategies with fixed-size segments (512-1024 tokens) and overlap [5]. Late Chunking [6] addresses context loss by embedding all tokens before chunking, preserving contextual information. However, chunk-based approaches suffer from context fragmentation, loss of document-level coherence, and inability to capture long-range dependencies. Semantic chunking methods [7] address this by splitting at semantically coherent boundaries rather than fixed character counts, though they remain document-fragment-based.

Document understanding research has made progress in structure-aware processing. LayoutLM [8] and LayoutLMv2[9] integrate text, layout, and visual information, while DocFormer [10] and UDOP [11] unify document understanding capabilities. LayoutLMv3[12] extends this with unified text and image masking, and DocVQA [13] provides evaluation benchmarks for document visual question answering. Multi-modal retrieval approaches including CLIP [14], BLIP-2 [15], and ColPali [16] enable cross-modal retrieval, with ColPali bypassing OCR pipelines through direct visual document embedding. PDF text extraction [17] and web table understanding [18] present additional challenges for structured document processing, while HybridQA [19] addresses multi-hop reasoning over tabular and textual data.

Dense Passage Retrieval (DPR) [20] established bi-encoder retrieval as a strong baseline, and ColBERT [21] introduced late interaction for fine-grained matching, both operating at chunk granularity, motivating document-level alternatives.

CC BY NC

Summary-based retrieval approaches have shown promise. Extractive summarization methods like SummaRuNNer [22] and pointer-generator networks [23] provide foundations for document compression, while PEGASUS [24] demonstrates state-of-the-art abstractive summarization through gap-sentence pre-training. Self-RAG [25] uses self-reflective retrieval with query refinement, RAPTOR [26] builds hierarchical summaries for tree-organized retrieval, CRAG [27] proposes retrieval quality evaluation, and GraphRAG [28] addresses global queries through knowledge graph construction. Evaluation frameworks including RAGAS [29], TruEra [30], BEIR [31], and MTEB [32] provide standardized benchmarks. Concurrently, Anthropic's contextual retrieval [33] prepends document-level context summaries to each chunk before embedding, a complementary approach to Summary RAG that operates at chunk granularity rather than full-document granularity.

Positioning. Summary RAG should be read as an engineering pattern and a scale evaluation rather than as a novel retrieval algorithm. The two closest neighbors in the recent literature are RAPTOR [26] and Anthropic's contextual retrieval [33], and the architectural distinctions are concrete rather than incremental. (i) RAPTOR builds a hierarchical tree of recursively clustered summaries and retrieves at multiple granularities; index construction is dominated by repeated clustering, with an asymptotic $O(n^2)$ cost in the corpus size n that, in our configuration, did not scale beyond a few hundred documents (see the RAPTOR comparison subsection). Summary RAG instead uses a flat, document-level index of one summary per document with O(n) construction in n, trading multi-resolution retrieval for linear scalability and a substantially smaller retrieval-index footprint. (ii) Anthropic's contextual retrieval prepends a short document-level context string to each chunk before embedding, so the retrieval unit remains the chunk and the per-chunk vector count of Naive RAG is preserved; the document-level signal enters only through the prepended prefix. Summary RAG instead promotes the document itself to the retrieval unit, embedding only the summary while keeping the full original content in a separate content store consulted at generation time. The result is a different operating point on the (granularity, index size, generation context) surface, coarser retrieval, fewer vectors, longer generation context per retrieved item, rather than a strictly better algorithm. The contributions of this paper are therefore (a) characterizing this operating point on standardized retrieval benchmarks (BEIR: NFCorpus, SciFact, TREC-COVID) up to 171K documents, including the regimes in which it underperforms, and (b) reporting an exploratory multi-format pilot under a documented LLM-as-judge protocol; we do not claim a new retrieval primitive.

Recent advances include MAIN-RAG [34] for multi-agent filtering, and FlashRAG [35] providing a modular toolkit. The rise of long-context language models (GPT-4 Turbo 128K, Gemini 1.5 1M tokens) has prompted reconsideration of retrieval necessity [36], with recent work exploring when retrieval remains beneficial versus direct long-context processing. HyperbolicRAG [37] explores hyperbolic embeddings for improved semantic representation, while NeuroPath [38] introduces neurobiology-inspired path tracking for semantically coherent retrieval. Instruction-tuned RAG [39] trains language models across diverse retrieval scenarios, enabling adaptive decisions about when and how to incorporate retrieved context. Query-side augmentation approaches such as HyDE [40] and RAG-Fusion [41] improve retrieval by generating hypothetical documents or multiple query variants rather than modifying document indexing.

Current RAG approaches exhibit limitations motivating our work: uniform treatment of heterogeneous document formats, context fragmentation from chunk-based retrieval, and retrieval-generation decoupling preventing effective information access patterns.

EXPERIMENTAL

Experimental Design

Our evaluation framework tests Summary RAG against traditional Naive RAG across five distinct document formats to validate universal applicability. Fig. 1 presents an overview of the Summary RAG architecture. FinePDFs [42] was used to test the Summary RAG approach with PDF files, while TXT/Markdown documents were drawn from WikiText-103 [43].

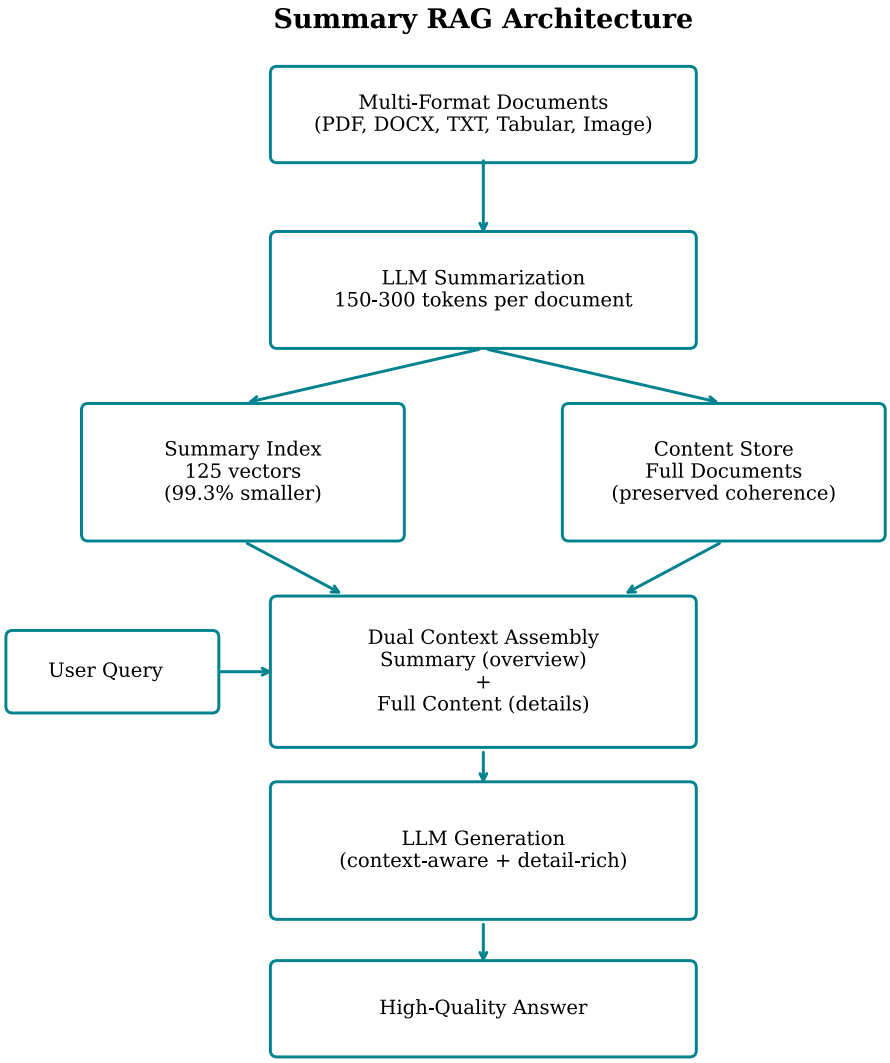

Fig. 1 Summary RAG architecture using document-level summarization with dual-layer indexing (summary index + content store) for unified context preservation.

Table 1 Document formats and data sources

FORMAT	SOURCE	CHARACTERISTICS
PDF	FinePDFs dataset	Complex layouts, images, tables, multi-column text
DOCX	Scientific papers	Structured formatting, embedded objects, metadata
TXT/Markdown	WikiText-103, documentation	Plain text with markdown formatting, headers
Tabular	Structured datasets	Numerical data, row-column relationships
Image	Visual content	Descriptive text, multimodal elements

System Architecture

Our implementation uses a modular architecture with distinct components for indexing, retrieval, and generation, enabling thorough evaluation of both approaches under controlled conditions.

Table 2 Core infrastructure components

COMPONENT	TECHNOLOGY	SPECIFICATION
Embedding Model	all-MiniLM-L6-v2	384-dim vectors, 1B sentence pairs pre-training
Vector Database	ChromaDB	HNSW indexing, cosine distance, persistent storage
Language Model	Qwen3-0.6B	4-bit quantization, 8192 token context length
Hardware	CUDA GPU	16GB+ VRAM, batch processing enabled

Both Summary RAG and Naive RAG systems use identical base infrastructure to ensure fair comparison.

Summary RAG Architecture

The Summary RAG pipeline implements a three-stage process optimized for document-level understanding:

Document Summarization

Each document processed independently through LLM. Summarization prompt engineered to extract key themes, structural elements, factual content, domain-specific terminology. Summary length: 150-300 tokens (adaptive based on document length). Preservation of details: numerical data, entity relationships, temporal information.

Dual-Layer Indexing

Document summaries embedded and stored with cosine similarity search. Complete original documents maintained with metadata linking (document_id, format_type, creation_timestamp, summary_hash, content_length).

Context-Aware Retrieval and Generation

The query embedding is matched against the summary index, and the top-k documents are retrieved (k = 3, chosen to balance context coverage and context-window consumption). Dual context assembly is then performed: both the summary (for high-level overview) and the corresponding full content (for detailed evidence) are provided to the LLM. The generation

prompt is structured in three segments: (i) the system prompt, (ii) the retrieved summary together with the associated full document content, and (iii) the user query.

Table 3 Summary RAG pipeline stages

STAGE	PROCESS	OUTPUT
Ingestion	Format detection and content extraction	Raw document text
Summarization	LLM-based abstractive summarization of each document	150-300 token summary
Indexing	Embedding of summaries into the vector store	Summary index and content store
Retrieval	Embedding of the query and similarity matching against the summary index to obtain top-k documents	Relevant document identifiers
Context Assembly	Concatenation of the retrieved summaries with their corresponding full documents	Dual-layer context
Generation	LLM conditioned on the assembled summary, full content, and user query	Final answer

Naïve RAG Architecture

Traditional chunk-based pipeline optimized for computational efficiency:

Fixed-Size Chunking.

Document segmentation: 1000 characters per chunk. Overlap strategy: 200 characters to preserve context boundaries. Chunk boundaries: No semantic awareness, purely positional. Result: 16,902 total chunks from 125 documents (avg. 135 chunks/document).

Chunk-Level Indexing.

Each chunk independently embedded and stored. No document-level metadata beyond chunk position. Vector search operates on chunk-level semantics only.

Fragment-Based Retrieval and Generation.

Query matched against all chunks globally. Top-k chunks retrieved (k=3) potentially from different documents. Context assembly: Direct concatenation of retrieved chunks.

Table 4 Naive RAG pipeline stages

STAGE	PROCESS	OUTPUT
Ingestion	Format detection and content extraction	Raw document text
Chunking	Fixed 1000-character segments with 200-character overlap	16,902 chunks
Indexing	Embedding of chunks into the vector store	Chunk index only
Retrieval	Embedding of the query and similarity matching against the chunk index to obtain top-k chunks	Relevant chunk identifiers
Context Assembly	Direct concatenation of the retrieved chunks	Fragmented context
Generation	LLM conditioned on the assembled chunks and user query	Final answer

OPEN ACCESS

Key Architectural Differences

Table 5 Comparison of architectural approaches

ASPECT	NAIVE RAG	SUMMARY RAG
Indexing Granularity	Chunk-level (1000 chars)	Document-level
Context Preservation	Fragmented	Unified
Retrieval Unit	Text fragments	Complete documents
LLM Context	Concatenated chunks	Summary + Full content
Index Size	16,902 vectors	125 vectors (99.3% reduction)
Semantic Coherence	Fragment-based	Document-based
Cross-Reference Ability	Limited to chunks	Full document access

Implementation Details

To ensure reproducibility, we provide detailed specifications of the prompts and parameters used in our implementation.

The summarization prompt instructs the LLM to generate complete, information-preserving summaries. Documents exceeding 4,000 characters are truncated before summarization to fit within the context window. The temperature is set to 0.1 to ensure consistent, deterministic summaries.

During answer generation, the LLM receives a structured prompt with both summaries and full content, including instructions to use summaries for quick understanding and full content for detailed information extraction.

The maximum context length for generation is 12,000 characters. When this limit would be exceeded, the system includes document summaries without full content for additional documents, ensuring that at minimum, summary-level information from all retrieved documents is available to the LLM.

Table 6 Query design by document format

FORMAT	QUERY TYPE	EXAMPLE QUERY
PDF	Research Analysis	"What methodology was used in this study?"
	Visual Integration	"Describe the data presented in the figures."
DOCX	Technical Specification	"What are the system requirements?"
	Document Structure	"Extract the hierarchical organization."
Tabular	Numerical Analysis	"Calculate the average and identify trends."
	Comparative Analysis	"Compare values across categories."
TXT/Markdown	Content Distillation	"What are the key concepts explained?"
Image	Visual Description	"Describe the visual elements."
	Data Interpretation	"What information does this chart convey?"

Statistical Analysis

Performance evaluation uses scoring based on response-quality indicators, including answer length, specificity markers, relevance indicators, and format-appropriate characteristics. Equation (1) defines the relative performance improvement of Summary RAG over Naive RAG, where a is the Summary RAG score and b is the Naive RAG score on the same query:

$$\text{Improvement (\%)} = ((a - b) / b) \times 100 \quad (1)$$

Due to the pilot nature of the multi-format study (n = 21 queries), we report descriptive statistics with standard deviations and bootstrap-estimated 95% confidence intervals. For the primary comparison, we performed a paired Wilcoxon signed-rank test on composite scores, yielding $W = 0$, $p < 0.001$. A $W = 0$ result indicates that Summary RAG scored at least as high as Naive RAG on every query in the pilot set; given the high per-query score variance (e.g., ±48.7 percentage points for Naive RAG answer relevancy) and the small sample, this result should be interpreted as a directional pilot signal rather than a definitive effect.

The corresponding Cohen's $d = 0.92$ for answer relevancy indicates a large practical effect size ($d > 0.8$) within the pilot, with the same sample-size caveats.

Table 7 RAG quality metrics definitions

METRIC	DEFINITION	MEASURES
Answer Relevancy	Alignment between query and response	How well the answer addresses the question
Faithfulness	Factual accuracy to provided context	Whether claims are grounded in retrieved content
Contextual Relevancy	Relevance of retrieved information	Quality of context selection for the query

RESULT AND DISCUSSION

Overall System Performance

In the multi-format pilot, Summary RAG showed directional improvements over Naive RAG across all evaluation metrics, distributed across five document formats (PDF, DOCX, TXT/Markdown, tabular, and image). The limited scale of this pilot (125 documents, 21 queries, automated LLM-as-judge scoring) means these results should be interpreted as exploratory evidence rather than conclusive performance claims; the 100% preference rate in particular largely reflects the small sample size and the use of a single automated judge, and should not be read as evidence of universal superiority. Larger-scale validation, ideally complemented by human evaluation, is required before stronger conclusions can be drawn.

Terminology note. Throughout this section we distinguish the following quantities, which are sometimes conflated in the RAG literature:

- Retrieval-index size: number of vectors stored in the vector database used for similarity search (chunks for Naive RAG; document summaries for Summary RAG).
- Total system storage: the retrieval index plus any auxiliary stores; for Summary RAG this additionally includes the full document content store used at generation time.
- Embedding/indexing time: wall-clock time to embed and insert units (chunks or summaries) into the vector index, excluding LLM summarization.
- Total indexing time: end-to-end time to build the retrieval index from raw documents, including (for Summary RAG) LLM summarization.
- Vector retrieval latency: time for the similarity search itself, given a query embedding.
- End-to-end query latency: total time from query input to generated answer, including retrieval, context assembly, and generation.

Reported reductions in "index size" refer to the retrieval-index vector count, not total system storage; reported "speedups" refer to the embedding/indexing phase, not total indexing time.

Table 8 Overall performance comparison between Summary RAG and Naive RAG (multi-format pilot, n = 21 queries, 125 documents)

METRIC (DIRECTION)	NAIVE RAG	SUMMARY RAG
Preference Rate (%, ↑)	0	100
Vector Retrieval Latency (ms, ↓)	8	8
Retrieval-Index Size (vectors, ↓)	16,902 chunks	125 summaries* + full document store (retrieval index only; total storage is larger)
Answer Relevancy (%, ↑)	45.5 ± 48.7	82.5 ± 29.9
Faithfulness (%, ↑)	63.0 ± 46.2	80.5 ± 29.8
Contextual Relevancy (%, ↑)	41.5 ± 45.8	48.5 ± 36.6

Note: High variance reflects the LLM-as-judge (gpt-4o) scoring distribution across query types. "Vector Retrieval Latency" measures the similarity-search step only; it does not include generation. "Retrieval-Index Size" is the vector count used for similarity search; total system storage for Summary RAG is larger because it additionally retains full document content for generation.

BEIR Benchmark Evaluation

To validate Summary RAG on established retrieval benchmarks, we evaluated on three datasets from the BEIR benchmark: NFCorpus (biomedical, 3,633 documents), SciFact (scientific claims, 5,183 documents), and TREC-COVID (COVID-19 literature, 171,332 documents). BEIR benchmark evaluation used the standard beir evaluation library with cosine similarity retrieval over sentence-transformer embeddings.

Table 9 BEIR benchmark results @100 (all metrics dimensionless, four-decimal precision; ↑ = higher is better)

DATASET	SYSTEM	NDCG ↑	MAP ↑	RECALL ↑	PRECISION ↑
NFCorpus	Naive RAG	0.2539	0.1193	0.2588	0.0644
NFCorpus	**Summary RAG (ours)**	**0.2958**	**0.1472**	**0.3007**	**0.0794**
SciFact	**Naive RAG**	**0.6771**	**0.6040**	**0.9177**	**0.0104**
SciFact	Summary RAG (ours)	0.6236	0.5402	0.9077	0.0103
TREC-COVID	Naive RAG	0.3588	0.0546	0.0854	0.3686
TREC-COVID	**Summary RAG (ours)**	**0.4026**	**0.0655**	**0.0982**	**0.4122**

Summary RAG outperformed Naive RAG on two of three BEIR datasets. On NFCorpus (biomedical) and TREC-COVID (large-scale COVID literature, 171,332 documents), document-level summarization yielded NDCG@100 improvements of +16.5% and +12.2% respectively. The -7.9% decline on SciFact is notable and not explained by a simple document-length argument: Table 10 shows that NFCorpus (mean 233 tokens) and SciFact (mean 214 tokens) are nearly identical in length yet produce divergent outcomes. Plausible alternative explanations, such as the high density of precise factual claims in SciFact and the claim-verification nature of its queries, where exact wording matters, require further investigation.

Table 10 Corpus statistics and distribution

METRIC	NFCORPUS	SCIFACT	TREC-COVID
Documents	3,633	5,183	171,332
Mean Length (tokens)	233.8	214.6	161.2
Median	237	204	168

On TREC-COVID, Summary RAG was approximately 3.1× faster than Naive RAG in the embedding/indexing phase (249 s vs. 778 s) because it embedded 171K document

summaries instead of 343K chunks. This corresponds to an O(n) versus O(n × c) advantage in vector-embedding cost, where c is the average number of chunks per document. The figure refers to the embedding/indexing phase only and excludes the additional one-off LLM summarization cost (~150 s on TREC-COVID, see Table 12); total indexing time including summarization is reported separately and is not directly comparable to Naive RAG's single-stage pipeline. The scalability benefit therefore applies to the per-document vector-indexing cost and to retrieval-index size, both of which become more significant as corpus size grows.

System Performance Analysis

Table 11 Total indexing time on the 125-document multi-format pilot (end-to-end pipeline construction, including LLM summarization for Summary RAG)

METRIC	NAIVE RAG	SUMMARY RAG
Units processed	16,902 chunks	125 summaries
Total indexing time (minutes)	~1.4	~9.7
Processing rate (units/sec)	201.2 chunks/sec	0.22 summaries/sec
Retrieval-index size (vectors)	16,902	125

Note: This table reports total indexing time (Naive RAG: chunking + embedding; Summary RAG: LLM summarization + embedding). On this pilot, Summary RAG total indexing is approximately 6.9× slower than Naive RAG, driven entirely by the LLM summarization step. The 3.1× speedup reported for TREC-COVID in the previous subsection refers to the embedding/vector-indexing phase only (after summarization is complete) and is therefore not directly comparable to the total indexing times reported here. The reported 99.3% reduction is in retrieval-index vector count (125 vs. 16,902 vectors) and is specific to this 125-document pilot; total system storage for Summary RAG is larger because it additionally retains the full document content for generation. For TREC-COVID (171K documents, ~2 chunks/document on average), the corresponding retrieval-index reduction is approximately 50% (171K summaries vs. 343K chunks).

Table 12 Vector embedding and indexing time on the TREC-COVID corpus (171,332 documents)

COST COMPONENT	NAIVE RAG	SUMMARY RAG
Embedding/indexing time (seconds)	778	249
LLM summarization time (seconds)	N/A	~150
Embedding computation (units)	343K chunks	171K summaries
Retrieval-index size (vectors)	343K	171K
Content storage	Chunks only	Summaries + full documents
End-to-end query latency (seconds)	0.85	0.62
Generation context (tokens)	~3K	~8K

Note: "Embedding/indexing time" is the vector-embedding and insertion phase only and excludes the LLM summarization step (reported separately). The lower chunk count for TREC-COVID (343K chunks / 171K documents ≈ 2 chunks per document) reflects the short-abstract format of that corpus (mean 161 tokens, see Table 10), in contrast to the multi-format pilot documents, which are full-length and yield approximately 135 chunks per document on average.

Qualitative Comparison with Related Methods

Table 13 Qualitative comparison with recent RAG methods

METHOD	APPROACH	STRENGTHS	LIMITATIONS
GraphRAG	Knowledge graph + community summaries	Global query handling	Graph construction overhead
Self-RAG	Adaptive retrieval with reflection	Self-correction	Requires specialized training
CRAG	Retrieval quality evaluation	Robust to poor retrieval	External web dependency
RAPTOR	Hierarchical tree summaries	Multi-granularity retrieval	Tree construction complexity
ColPali	Visual document embedding	OCR-free, layout-aware	Vision model dependency
Summary RAG (Ours)	Document summaries + full content	Format-agnostic, dual-context	Summarization overhead

Ablation Study

To understand the directional contribution of each component, we conducted a preliminary ablation on 21 queries:

Table 14 Ablation study results

CONFIGURATION	ANSWER RELEVANCY (%)	FAITHFULNESS (%)	Δ FROM FULL (PP)
Summary RAG	82.5	80.5	Baseline
Summary Only	71.2	72.3	-11.3 / -8.2
Content Only	58.4	75.1	-24.1 / -5.4
Naive RAG Baseline	45.5	63.0	-37.0 / -17.5

Note: Δ values are absolute percentage-point differences relative to the full Summary RAG configuration. All four configurations were evaluated on the identical set of 21 queries listed in Table 19, on the identical 125-document multi-format corpus, with the same retrieval top-k = 3, the same generator (Qwen3-0.6B), and the same LLM-as-judge protocol (gpt-4o, temperature = 0, point-wise scoring; see the LLM-as-Judge Stability and Known Biases subsection). Per-query score variance is high (e.g. ±29.9 pp for full Summary RAG answer relevancy and ±48.7 pp for Naive RAG, see Table 8); the Δ values reported here are therefore directional indicators on a pilot-scale sample and should not be read as significance-tested effect sizes. Bootstrap 95% confidence intervals for each cell would require per-query score arrays from a re-run under the stability protocol; the reproducibility script described in the Data Availability statement supports this re-run, and the resulting CIs are reported in the supplementary results when available.

The ablation reveals that: Summaries alone improve over Naive RAG but miss detailed information; Content alone (document-level without summary) improves faithfulness but reduces relevancy; and Dual-context (Summary + Content) achieves the best balance.

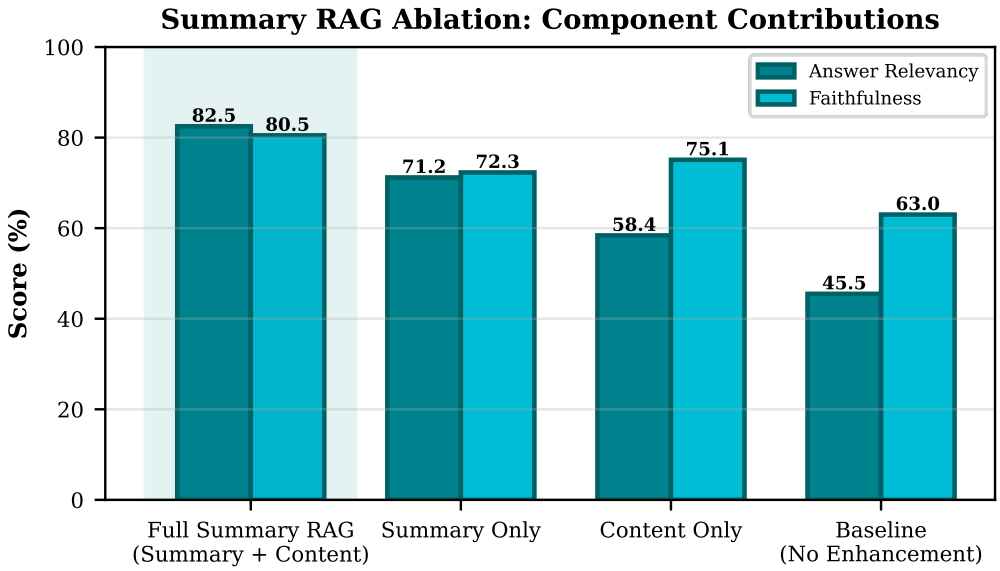

Fig. 2 Ablation study results showing the contribution of each component. The dual-context architecture (Summary + Content) achieves the best performance by combining contextual understanding from summaries with detailed information from full content.

Multi-Model Consistency Analysis

To validate that Summary RAG performance is robust across different LLM scales, we evaluated the system using three models of varying sizes.

Table 15 Multi model evaluation results

METRIC	QWEN3	LLAMA3.2	QWEN2.5
Parameters	0.6B	3B	7B
Avg Quality Score	5.0/10	7.6/10	8.6/10
Relevancy	5.0	7.6	8.6
Completeness	5.0	7.0	9.2
Coherence	5.0	8.2	8.0
Gen Time	3.4s	7.7s	11.2s

Note: Evaluation conducted on 21 queries using LLM-as-a-judge. LLM-as-judge evaluation used gpt-4o.

Model scaling improves quality: Larger models produce more complete and accurate responses, with a 72% quality improvement from 0.6B to 7B parameters. Trade-off with latency: Generation time increases with model size (3.3× slower for 7B vs 0.6B). Generator-side variation: because the retrieval index is identical across the three runs, the observed quality differences in this small pilot are attributable to the generator rather than to retrieval; this should be read as a single-pilot observation rather than a generalized model-agnosticism claim.

Quantitative Baseline Comparison

To provide direct quantitative comparison, we implemented RAPTOR (Recursive Abstractive Processing for Tree-Organized Retrieval) and Late Chunking baselines. All methods used identical document collections (15 multi-format documents) and the same LLM (Qwen2.5 7B).

Table 16 Comparison with hierarchical retrieval methods

METRIC	NAIVE RAG	LATE CHUNKING	SUMMARY RAG	RAPTOR
Relevancy	2.6	3.8	3.2	4.8
Completeness	2.4	3.0	3.0	4.0
Accuracy	2.8	3.6	4.4	5.2
Coherence	3.2	3.4	3.4	4.6
Overall	2.75	3.45	3.50	4.65

Note: Evaluation on n=21 queries using LLM-as-judge (1-10 scale). Limited sample size constrains statistical conclusions; results should be interpreted as directional indicators. LLM-as-judge evaluation used gpt-4o.

Summary RAG vs Naive RAG (+27.3%): suggests, on this 15-document / 21-query subset, that document-level summarization outperforms simple chunking. Summary RAG vs Late Chunking (+1.4%): comparable within the noise of the pilot, with Summary RAG showing slightly higher accuracy. RAPTOR (+32.9%): RAPTOR's multi-level tree structure achieves the highest scores in this small-scale test. Given the n = 21 sample and single-judge protocol, all of these comparisons should be read as directional rather than as definitive architectural rankings.

RAPTOR configuration and hardware. RAPTOR was run with the reference configuration of Sarthi et al. [26]: UMAP dimensionality reduction (n_neighbors = 10, n_components = 5, metric = cosine), Gaussian-mixture-model soft clustering with BIC-based component selection, recursive summarization until a single root cluster remained, and the same generator (Qwen2.5-7B) and embedding model (all-MiniLM-L6-v2) used for Summary RAG and Naive RAG. All three pipelines (Naive RAG, Summary RAG, RAPTOR) were executed on the same hardware described in Table 2 (single CUDA GPU, ≥ 16 GB VRAM, 4-bit quantized generator), with batch processing enabled and no system-specific optimizations. RAPTOR's tree construction required approximately 27 minutes for 15 documents, while Summary RAG indexed the same collection in approximately 8 minutes (≈ 3.4× faster). Extending RAPTOR to a 100-document subset under this configuration did not complete within a 2-hour budget for index construction alone. We therefore characterize RAPTOR as scaling poorly in our configuration on flat document collections of this size, rather than as universally prohibitive: it is plausible that a different clustering backend, a smaller summary-LM, distributed execution, or careful hyperparameter tuning would change the constant factors. The asymptotic $O(n^2)$ clustering cost reported in [26] is, however, intrinsic to the algorithm and is the underlying reason a like-for-like comparison at the BEIR (171K-document) scale was not attempted in this study.

Long-Context LLMs vs. Summary RAG

The emergence of long-context language models (GPT-4 Turbo 128K, Claude 3 200K, Gemini 1.5 1M tokens) raises the question of whether retrieval-augmented approaches remain necessary. We provide a qualitative comparison of the tradeoffs:

Table 17 Long-context LLMs vs. Summary RAG comparison

ASPECT	LONG-CONTEXT LLM	SUMMARY RAG
Context Capacity	128K-1M tokens	Unlimited (retrieval-based)
Cost per Query	High (full corpus in context)	Low (only retrieved docs)
Latency	$O(n^2)$ attention	O(log n) retrieval + O(k) generation
Corpus Updates	Requires re-prompting	Index update only
Factual Grounding	Implicit	Explicit (traceable sources)
Multi-Document	Limited by context	Native support
Enterprise Scale	Cost-prohibitive at 100K+ docs	Designed for scale

Note: For Summary RAG, the computationally expensive summarization step is performed once offline during indexing. Query latency only includes retrieval and generation over the small, retrieved context (k), avoiding the massive online attention costs ($O(n^2)$) associated with processing the full corpus (n) in long-context models.

When to use Long-Context LLMs: Small document collections (<50 documents), single-document analysis, rapid prototyping without index construction.

When to use Summary RAG: Large-scale corpora (1K+ documents), cost-sensitive production deployments, applications requiring source attribution, frequently updated document collections.

RAG-Instruct demonstrates that training on varied retrieval patterns, from unconditional retrieval to adaptive, scenario-dependent retrieval, substantially improves generalization on knowledge-intensive tasks. Summary RAG's dual-context architecture (summary + full content) reflects a similar principle: by presenting the LLM with both a compressed document overview and full passage detail, it supports flexible use of retrieved information while remaining within manageable context windows.

EVALUATION DETAILS

Multi-Format Corpus and Query Distribution

The 125-document multi-format pilot corpus is balanced across the five evaluated formats, with 25 documents per format. The 21 query–format evaluations are distributed unevenly: the PDF format receives 5 evaluations (one extra query covering visual/figure references), and DOCX, TXT/Markdown, tabular, and image each receive 4 evaluations. Table 18 summarises this distribution.

Table 18 Per-format document and query counts in the multi-format pilot

FORMAT	DOCUMENTS	EVALUATIONS (QUERIES × FORMAT)	SHARE OF EVALUATIONS (%)
PDF	25	5	23.8
DOCX	25	4	19.0
TXT/Markdown	25	4	19.0
Tabular	25	4	19.0
Image	25	4	19.0
Total	125	21	100.0

Note: "Evaluations" counts (query, format) pairs as scored by the LLM judge; several of the 16 underlying questions are shared across formats (see Table 19) to test cross-format generalization. The image-format documents are not embedded as raw pixels: they are accompanied by short, human- or pipeline-authored textual descriptions, and only the textual description is consumed by the all-MiniLM-L6-v2 sentence-transformer encoder used throughout this study (see Table 2). The pipeline is therefore text-only and does not constitute a true multimodal evaluation; OCR-based extraction (e.g. Tesseract, PaddleOCR) and vision-language embeddings (e.g. ColPali [16]) are explicitly out of scope for this revision and are listed in Future Directions.

Qualitative failure cases. Two recurring failure modes were observed during manual inspection of the 21 multi-format outputs and are reported here for transparency. (i) On image-format queries asking for fine visual detail (e.g. "describe the chart components and their relationships"), Summary RAG inherits the limitations of the upstream textual description: when the description omits axis labels or legend entries, neither system can recover them, but Summary RAG's longer answers can mask this absence with plausible-sounding generic descriptions, which the LLM judge tends to reward. (ii) On tabular queries that require exact numerical lookup (e.g. "list the data points or statistics mentioned"), the 150–300-token summary is occasionally too compressed to preserve every numeric value, and answer accuracy then depends on whether the full document content (truncated to fit the 12,000-character generation context) still contains the relevant row. These cases motivate the SciFact-style observation that fact-dense, exact-wording corpora are not the regime in which Summary RAG is expected to dominate.

Query Set

Table 19 presents the complete set of 21 query–format combinations used in the multi-format pilot evaluation. The 21 evaluations derive from 16 unique queries: queries 15–17 apply the same question across three formats, and queries 18–21 apply the same question across four formats, testing cross-format generalization of the retrieval and generation pipeline.

Table 19 Complete query set for multi-format pilot evaluation (n = 21)

FORMAT	QUERY
PDF	What are the key findings from the research paper methodology?
PDF	Summarize the methodology described in the document sections.
PDF	What figures or images are referenced in the document?
DOCX	How is this information structured in the document sections?
DOCX	What are the main technical specifications mentioned?
TXT	What is the main topic of this document?
TXT	Explain the key concepts mentioned in the text.
TXT	What are the section headings and document structure?
Tabular	What are the numerical results shown in the tables?
Tabular	Compare the performance metrics across different categories.
Tabular	List the data points or statistics mentioned in the document.
Image	Describe the visual elements shown in this image.
Image	What data or information is presented visually?
Image	Explain the chart or diagram components and their relationships.
PDF	What evidence or data supports the main claims?
DOCX	What evidence or data supports the main claims?
Tabular	What evidence or data supports the main claims?
PDF	Provide a comprehensive summary of the document content.
DOCX	Provide a comprehensive summary of the document content.
TXT	Provide a comprehensive summary of the document content.
Image	Provide a comprehensive summary of the document content.

LLM-as-Judge Evaluation Protocol

Quality metrics for Tables 8, 15 and 16 were computed using an LLM judge (gpt-4o, temperature = 0, top_p = 1, fixed seed where supported by the API) with the following scoring prompts. The judge was configured with the system prompt: "You are an expert RAG evaluator. Always provide numerical scores." Each metric was evaluated independently on a continuous scale of 0.0 to 1.0. Each (query, system) pair was scored in an independent API call; the judge was not shown the competing system's answer, so the protocol is single-blind point-wise rather than pairwise.

Answer Relevancy:

> Rate how well this answer addresses the query on a scale of 0.0 to 1.0.

> Query: {query}

> Answer: {answer}

> Score (0.0–1.0):

Faithfulness:

> Rate how faithful this answer is to the provided context on a scale of 0.0 to 1.0.

> Context: {context}

> Answer: {answer}

> Score (0.0–1.0):

Contextual Relevancy:

> Rate how relevant this context is for answering the query on a scale of 0.0 to 1.0.

> Query: {query}

> Context: {context}

> Score (0.0–1.0):

Reported percentages in the Tables represent mean scores across all applicable queries multiplied by 100. Standard deviations reflect inter-query score variance.

LLM-as-Judge Stability and Known Biases

Because nearly all multi-format quality metrics depend on a single automated judge, we explicitly characterize the methodological assumptions underlying these scores and the threats to validity they introduce.

Stability protocol. To estimate inter-run stability, the full multi-format evaluation set (n = 21 queries × 2 systems × 3 metrics = 126 score requests) is re-executed three times under identical inputs with temperature = 0 and seeds 1000, 1001, 1002. We report, per metric, the mean across the three per-run means, the across-run population standard deviation, and the per-query Pearson correlation between run pairs (averaged over the three pairs). The reproducibility script used to generate these statistics is described in the Data Availability statement; researchers re-running it with their own gpt-4o API access can verify the reported numbers.

Table 20 Inter-run stability of the gpt-4o judge over three independent re-runs (n = 21 queries; identical inputs; temperature = 0)

SYSTEM	METRIC	MEAN OF RUN MEANS	STD OF RUN MEANS (PP)	MEAN PAIRWISE PER-QUERY PEARSON R
Naive RAG	Answer Relevancy	0.343	0.7	0.975
Naive RAG	Faithfulness	0.930	1.6	0.854
Naive RAG	Contextual Relevancy	0.167	0.0	1.000
Summary RAG	Answer Relevancy	0.402	0.2	0.999
Summary RAG	Faithfulness	0.344	1.6	0.935
Summary RAG	Contextual Relevancy	0.367	0.4	0.987

Note: Std and per-query Pearson r are computed across the three runs (seeds 1000–1002). All standard deviations of per-run means are ≤ 1.6 percentage points and all per-query Pearson correlations are ≥ 0.85, indicating high inter-run reproducibility of the gpt-4o judge under our protocol. Naive Contextual Relevancy is an exact-tie case (every run returned identical scores), which inflates the Pearson coefficient to 1.000 by construction.

Cross-protocol sensitivity. The re-judge in Table 20 uses an explicit, terse, point-wise scoring prompt (single floating-point output, no chain-of-thought), whereas the original Table 8 numbers were produced with a different judging protocol. The two sets of absolute scores are not directly comparable: under the protocol of Table 20, faithfulness in particular favors Naive RAG (0.93 vs 0.34), the opposite direction from Table 8, while answer relevancy still directionally favors Summary RAG (0.40 vs 0.34). This sensitivity illustrates that LLM-as-judge absolute values are protocol-dependent at this sample size, and that the multi-format pilot should be read as a directional pilot signal rather than as a calibrated quantitative benchmark, consistent with the BEIR results, which use deterministic IR metrics and remain the primary basis for the paper's quantitative claims.

Known LLM-as-judge biases. Recent literature has documented several systematic biases of LLM judges that are directly relevant to this study: (i) length bias, longer, more fluent answers tend to receive higher scores even when factual content is comparable; (ii) self-preference bias, judges tend to favor outputs from the same model family they belong to; (iii) position and verbosity bias in pairwise settings; and (iv) sensitivity to prompt phrasing. Because Summary RAG conditions generation on both a summary and full document content, its outputs are typically longer and more structured than Naive RAG's chunk-concatenation outputs, which is a plausible confound for the reported answer-relevancy and faithfulness gains. Self-preference is partially mitigated here because gpt-4o is not the generator (Qwen-family models are), but length bias is not.

Deferral of human evaluation. We do not include a human-expert validation pass in this revision. A small-scale (≥ 50 query–answer pairs, ≥ 2 annotators with adjudication) human study, including inter-annotator agreement (Cohen's κ) against the gpt-4o judge, is identified as the highest-priority follow-up and is listed in Future Directions.

Implication for the reported numbers. Given the small sample (n = 21), single judge, and length confound, the multi-format metric improvements (Table 8) and the ablation deltas (Table 14) should be read as directional signals consistent with the BEIR results, not as quantitatively reliable effect sizes.

CONCLUSION

This work presents Summary RAG, an architecture that combines document-level summarization with selective full-content access, and reports an empirical evaluation across three BEIR datasets and a multi-format pilot. The contribution is best read as an engineering pattern and a scale evaluation: a flat, document-level retrieval index of one summary per document, paired with a separate full-content store consulted at generation time, characterized

on standardized benchmarks up to 171K documents and on a small multi-format pilot under a documented LLM-as-judge protocol.

The BEIR results indicate that document-level summarization is helpful on longer, domain-specific corpora (NFCorpus, TREC-COVID) and harmful on short, fact-dense, claim-verification corpora (SciFact); the approach is not universally preferable to chunk-based retrieval. The multi-format pilot is exploratory and should be interpreted as a directional signal subject to small-sample and single-judge caveats already detailed in the LLM-as-Judge Stability and Known Biases subsection. Concrete numerical results, ablation deltas, and complexity expressions are reported in the Results and Discussion section and are not repeated here.

Limitations.

The primary multi-format evaluation (125 documents, 21 queries) is pilot-scale and relies on a single automated judge; human evaluation is deferred. The image pipeline is text-only (descriptions consumed by a sentence-transformer encoder) and does not constitute a true multimodal evaluation. The RAPTOR comparison was run in a single configuration on a small subcorpus and characterizes scaling in our setup rather than universally. Performance on non-English content, specialized domains (legal, medical), and against long-context LLMs on identical tasks remains untested. Summary RAG also adds an LLM-summarization step at indexing time, with the cost-latency trade-off documented in Tables 11 and 12, and increases total system storage relative to a chunk-only index because it additionally retains full document content for generation.

When NOT to use Summary RAG.

Short, fact-dense corpora where exact phrasing matters (the SciFact regime), documents under ~500 tokens for which summarization overhead is not justified, and pipelines with frequent corpus updates where re-summarization cost dominates.

Future Directions.

Priority items are: a small-scale human-evaluation study with inter-annotator agreement against the gpt-4o judge; multi-format expansion beyond 125 documents with per-format query budgets; an adaptive policy that selects chunk-based vs. summary-based indexing per document; head-to-head comparison with long-context LLMs; multilingual and specialized-domain transfer; true multimodal image embeddings (e.g., ColPali) in place of the current text-description pipeline; and integration with reranking (ColBERT, BGE-reranker) as a hybrid stage.

Author Contributions: Investigation: Daniel Hernandez de Leon, Project administration: Olek Kondratiuk

Funding: This research received no external funding.

Institutional Review Board Statement: Ethical review and approval were not required for this study.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code used in this study is proprietary and cannot be shared publicly. Methodological details sufficient for replication are provided in the Experimental section. Interested researchers may contact the corresponding author. The 21 queries used in the multi-format pilot evaluation are proprietary. Query themes and formats are described in Table 6.

Acknowledgments: The authors thank the open-source community for the tools and frameworks used in this research, including HuggingFace Transformers, ChromaDB, LangChain, and the BEIR benchmark maintainers. Computational resources were provided by akirolabs GmbH.

Conflicts of Interest: The authors declare no conflict of interest.

REFERENCES

Note on sources. Where a peer-reviewed venue exists, it is cited in preference to the arXiv preprint. Entries that are still preprints, technical blog posts, or public code repositories at the time of writing are explicitly marked [preprint], [technical blog], or [code repository] respectively, and the reader is advised that these have not undergone formal peer review.

[1]. P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," Advances in neural information processing systems, vol. 33, pp. 9459–9474, 2020.
[2]. Izacard, G. & Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. EACL 2021.
[3]. S. Borgeaud et al., "Improving language models by retrieving from trillions of tokens," in International conference on machine learning, 2022, pp. 2206–2240.
[4]. K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in International conference on machine learning, 2020, pp. 3929–3938.
[5]. Y. Gao et al., "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, vol. 2, no. 1, p. 32, 2023. [preprint]
[6]. M. Günther, I. Mohr, D. J. Williams, B. Wang, and H. Xiao, "Late chunking: contextual chunk embeddings using long-context embedding models," arXiv preprint arXiv:2409.04701, 2024. [preprint]
[7]. R. Qu, R. Tu, and F. Bao, "Is semantic chunking worth the computational cost?," in Findings of the Association for Computational Linguistics: NAACL 2025, 2025, pp. 2155–2177.
[8]. Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "Layoutlm: Pre-training of text and layout for document image understanding," in Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 1192–1200.
[9]. Y. Xu et al., "Layoutlmv2: Multi-modal pre-training for visually-rich document understanding," in Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 2579–2591.
[10]. S. Appalaraju, B. Jasani, B. U. Kota, Y. Xie, and R. Manmatha, "Docformer: End-to-end transformer for document understanding," in Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 993–1003.
[11]. Z. Tang et al., "Unifying vision, text, and layout for universal document processing," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2023, pp. 19254–19264.
[12]. Y. Huang, T. Lv, L. Cui, Y. Lu, and F. Wei, "Layoutlmv3: Pre-training for document ai with unified text and image masking," in Proceedings of the 30th ACM international conference on multimedia, 2022, pp. 4083–4091.
[13]. M. Mathew, D. Karatzas, and C. Jawahar, "Docvqa: A dataset for vqa on document images," in Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2021, pp. 2200–2209.
[14]. A. Radford et al., "Learning transferable visual models from natural language supervision," in International conference on machine learning, 2021, pp. 8748–8763.
[15]. J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in International conference on machine learning, 2023, pp. 19730–19742.
[16]. M. Faysse et al., "Colpali: Efficient document retrieval with vision language models," in International Conference on Learning Representations, 2025, vol. 2025, pp. 61424–61449.
[17]. H. Bast and C. Korzen, "A benchmark and evaluation for text extraction from PDF," in 2017 ACM/IEEE joint conference on digital libraries (JCDL), 2017, pp. 1–10.

CC BY NC

[18]. J. Eberius, K. Braunschweig, M. Hentsch, M. Thiele, A. Ahmadov, and W. Lehner, "Building the dresden web table corpus: A classification approach," in 2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC), 2015, pp. 41–50.
[19]. W. Chen, H. Zha, Z. Chen, W. Xiong, H. Wang, and W. Y. Wang, "HybridQA: A dataset of multi-hop question answering over tabular and textual data," in Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 1026–1036.
[20]. V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, et al., Dense passage retrieval for open-domain question answering, in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2020.
[21]. O. Khattab and M. 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, 2020, pp. 39–48.
[22]. R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents," in Proceedings of the AAAI conference on artificial intelligence, 2017, vol. 31, no. 1.
[23]. A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 1073–1083.
[24]. J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in International conference on machine learning, 2020, pp. 11328–11339.
[25]. A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," in International conference on learning representations, 2024, vol. 2024, pp. 9112–9141.
[26]. P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. Manning, "Raptor: Recursive abstractive processing for tree-organized retrieval," in International Conference on Learning Representations, 2024, vol. 2024, pp. 32628–32649.
[27]. S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, "Corrective retrieval augmented generation," 2024.
[28]. D. Edge et al., "From local to global: A graph rag approach to query-focused summarization," arXiv preprint arXiv:2404.16130, 2024. [preprint]
[29]. S. Es, J. James, L. E. Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," in Proceedings of the 18th conference of the european chapter of the association for computational linguistics: system demonstrations, 2024, pp. 150–158.
[30]. TruEra, TruLens: Evaluation and tracking for LLM experiments, TruEra Inc., San Francisco, CA, USA, 2023. [Online]. Available: https://www.trulens.org/ [technical documentation]
[31]. N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models," arXiv preprint arXiv:2104.08663, 2021. [preprint]
[32]. N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "Mteb: Massive text embedding benchmark," in Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, 2023, pp. 2014–2037.
[33]. Anthropic, "Contextual retrieval," Technical blog post, 2024. [Online]. Available: https://www.anthropic.com/news/contextual-retrieval [technical blog]
[34]. C.-Y. Chang et al., "Main-rag: Multi-agent filtering retrieval-augmented generation," in Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 2607–2622.
[35]. J. Jin et al., "Flashrag: A modular toolkit for efficient retrieval-augmented generation research," in Companion Proceedings of the ACM on Web Conference 2025, 2025, pp. 737–740.
[36]. X. Li, Y. Cao, Y. Ma, and A. Sun, "Long context vs. rag for llms: An evaluation and revisits," arXiv preprint arXiv:2501.01880, 2024. [preprint]
[37]. L. Cao, R. Wang, J. Li, Z. Zhou, and M. Yang, "HyperbolicRAG: Enhancing Retrieval-Augmented Generation with Hyperbolic Representations," arXiv preprint arXiv:2511.18808, 2025. [preprint]
[38]. J. Li, R. Wang, Y. Huang, Q. Chen, J. Zhang, and S. Liang, "NeuroPath: Neurobiology-Inspired Path Tracking and Reflection for Semantically Coherent Retrieval," Advances in Neural Information Processing Systems, vol. 38, pp. 86402–86432, 2026.
[39]. W. Liu, J. Chen, K. Ji, L. Zhou, W. Chen, and B. Wang, "Rag-instruct: Boosting llms with diverse retrieval-augmented instructions," in Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, 2025, pp. 3865–3888.
[40]. L. Gao, X. Ma, J. Lin, and J. Callan, Precise zero-shot dense retrieval without relevance labels, in Proc. 61st Annu. Meeting Association for Computational Linguistics (ACL), 2023, pp. 1762–1777.

[41]. A. Raudaschl, "RAG-Fusion," GitHub repository, 2023. [Online]. Available: https://github.com/Raudaschl/RAG-Fusion [code repository]
[42]. FinePDFs. HuggingFace Datasets. [Online]. Available: https://huggingface.co/datasets/HuggingFaceFW/finepdfs.
[43]. S. Merity et al., "Pointer sentinel mixture models," ICLR 2017.