Empirical research Study

Introduction

Motivation

Most commercial RAG systems use proprietary embedding and reranking models, such as Cohere Embeddings and Cohere Rerank 4. These deliver strong performance but keep their internal designs secret. Users treat them as black boxes.

Many RAG systems are built from proven, proprietary black-box components. This research follows that approach by fixing embeddings and rerankers, while highlighting the frequently neglected yet key significance of document chunking before retrieval. Chunking is not just another preprocessing step it actively shapes what information the retriever can access and directly determines recall and system effectiveness. Imagine a customer service system designed to extract answers from a large database of FAQs. Poor chunking can split crucial information across separate chunks, leading the system to miss vital answers or provide irrelevant ones. These practical failures not only result in dissatisfied customers and higher support costs, but also show that the way document chunking is handled fundamentally affects retrieval exactness and end-user satisfaction.

Chunking is essential because it defines the limits of the information the system can access and retrieve. Even the most advanced rerankers cannot recover relevant data if chunking fails to incorporate it in retrievable segments. Therefore, optimal chunking is not just foundational but vital to guaranteeing that critical content is retrievable. In this way, chunking ultimately controls whether the RAG system delivers effective results.

High-Level Problem

Most RAG pipelines follow a similar pattern:

1. Split documents into chunks ahead of time
2. Embed those chunks
3. Retrieve the most similar ones for a query.
4. Rerank the retrieved candidates.

The underlying assumption is that documents can be split into fixed, query-independent chunks and that embeddings with reranking will ensure subsequent relevance. Many tooling vendors recommend fixed-size chunks for faster deployment and greater simplicity. This belief persists in the industry as such a strategy permits predictable computing resources needed and easier implementation in production systems.

Rerankers improve precision by jointly processing the query and document. However, they only review retrieved chunks. If important information is split up, buried in large chunks, or needs nearby context, it may never reach the reranker. Chunking directly affects which information is available for reranking.

This creates a recall ceiling and raises a key question: Does fixed, query-independent chunking limit the performance of commercial RAG systems by restricting what advanced rerankers can access? This limitation could hurt businesses, such as a 30% drop in customer support efficiency due to missing information.

Research Questions

This research is guided by three questions:

- **RQ1:** When using commercial embedding and reranking APIs (Cohere Embed and Cohere Rerank 4), does semantic-aware chunking (LumberChunker) improve retrieval performance as opposed to standard fixed-size token chunking?
- **RQ2:** Are the gains from semantic chunking large enough to justify its higher preprocessing cost in a real system?
- **RQ3:** Does the impact of chunking depend on document characteristics like length, structure, or domain?

These questions separate two effects: how relevance is modeled (embeddings and reranking) versus how it is made retrievable (chunking).

Modern Reranking Architectures and Commercial APIs

Cross-Encoder Reranking in Research

Research demonstrates that cross encoder architectures outperform bi-encoder approaches for ranking tasks. Cross encoders process query-document pairs jointly using transformer models. This allows them to grasp subtle relevance signals. The Sentence Transformers documentation states, “Cross-encoders achieve better performances than bi-encoders because they perform attention across the query and the document” (Reimers and Gurevych).

Commercial Reranking APIs

Production RAG systems progressively leverage commercial reranking APIs, including Cohere Rerank 4. Such services offer cutting edge performance but operate as black boxes. Their internal architectures, training data, and optimization methods remain proprietary.

Commercial rerankers share several characteristics:

- state-of-the-art performance on commonly used benchmarks
- Large context windows (Cohere Rerank 4 supports 32K tokens)
- Proprietary architectures with undisclosed details
- Continuous model improvements and version updates

Cohere’s documentation states that Rerank 4 uses cross attention processing with a 32K token context window. It also includes automatic chunking for lengthy documents. However, details on its architecture, training methods, and implementation are proprietary.

Effects on Research Design

This research uses a systems evaluation approach. It measures the impact of chunking strategies on end-to-end retrieval performance with state-of-the-art commercial components. Rather than theorizing about internal mechanisms, it empirically measures what changes when chunking strategies vary. All other factors stay constant.

This method corresponds with common RAG development practices and delivers effective recommendations for practitioners. It deliberately excludes unfounded speculation about proprietary systems, ensuring empirical validity regardless of internal implementation details.

Existing Work: What Is Known About RAG Components

This section reviews current knowledge about retrieval-augmented generation systems. It focuses on the individual components: embeddings, retrieval strategies, reranking architectures, and document segmentation. It identifies where prior research has succeeded, the assumptions commonly made, and the major gaps that remain.

Dense Embeddings and Semantic Retrieval: Key Advances

Traditional information retrieval systems relied on lexical matching. BM25 was the dominant algorithm. Gao et al. note that while BM25 “remained state-of-the-art for decades and is still widely used today,” lexical retrieval “struggles when matching goes beyond surface forms and fails when query and document mention the same concept using different words (vocabulary mismatch)” (146).

Dense embedding retrieval emerged as an alternative that addresses vocabulary mismatch through the use of learned, meaningful embeddings. Gao et al. state, “embedding retrieval learns an encoder to understand and encode queries and documents. The encoded vectors can softly match beyond text surface form” (146).

The Core Compromise

Analyses consistently find that lexical and semantic retrieval methods have combined advantages and weaknesses. Gao et al. note, “Between these two models, one’s weakness is the other’s strength. Lexical retrieval performs exact token matching but cannot handle vocabulary mismatch. Meanwhile, embedding retrieval supports semantic matching but loses granular (lexical level) information” (147).

This complementarity has led to hybrid retrieval systems that combine both methods. Wang et al. found that dense retrievers effectively encode strong relevance signals but have difficulty with weaker ones a gap that can be addressed by combining them with BM25.

Modern production systems typically employ hybrid retrieval strategies that combine dense embeddings for semantic matching with BM25 for exact-term matching.

Reranking Architectures: From Bi-Encoders to Cross-Encoders

Architectural Differences

Research distinguishes performance differences between bi-encoder and cross-encoder architectures. Bi-encoders independently encode sentences into embeddings, which are then compared using cosine similarity. Cross encoders process both sentences through the transformer network and yield a relevance score. The key factor is cross-encoders’ ability to attend to both query and document, which captures nuanced conceptual associations (Reimers and Gurevych).

The trade-off is computing performance. Bi-encoders compress a document's possible meanings expressed in a single vector. This results in information loss but enables fast retrieval. Cross-encoders are more accurate but computationally expensive at scale.

The Two-Stage Retrieval Paradigm

Modern RAG systems typically use a dual-phase architecture:

1. Fast bi-encoder retrieval of top-k candidates (usually k=100-150)
2. Expensive cross-encoder reranking to improve top-n results (usually n=20)

This architecture balances computing performance with retrieval quality.

The Recall-Bounded Problem

A core constraint of reranking architectures has been identified in recent research. The final recall of a retrieve-and-rerank system is inherently limited by the underlying retriever. For example, the Recall@100 of a pipeline with a reranker cannot exceed the Recall@125 of the underlying retriever, regardless of reranker quality.

If relevant documents are not retrieved during the initial retrieval phase, they are permanently excluded from the final ranked list, regardless of how sophisticated the reranker is. This creates a mathematical ceiling on system performance.

Research examining ground-truth passages throughout several reranker budgets shows that although the percentage of returned answers does not always increase, the percentage of seen answers consistently grows with the reranker budget. This suggests that final precision of retrieval is limited not only by the retrieval algorithm's efficiency but also by discordance between the reranker's preferences and human-labeled relevance (Rathee et al.).

Document Chunking: The Understudied Component

Why Chunking Has Been Overlooked

Despite being a critical preprocessing step, document chunking has received relatively little research attention. Duarte et al. note, "One often overlooked part of the RAG pipeline is how textual content is segmented into 'chunks'. This can notably shape the dense retrieval quality" (6473).

Barnett et al. point out that "Software engineers face design decisions around how best to chunk the document and how large a chunk should be. If chunks are too small, certain questions cannot be answered. If the chunks are too long, then the answers include generated noise" (195).

Common Chunking Strategies

Static Token-Based Chunking: The default approach in production systems is to split documents into fixed-length token boundaries (e.g., 512 or 1024 tokens), sometimes with overlapping windows. Fixed-size chunking offers simplicity and predictable chunk sizes. It has key weaknesses: imprecise control over context size, risk of cutting words or sentences mid-thought, and lack of semantic consideration.

Semantic Chunking: Alternative approaches attempt to respect semantic boundaries. Semantic chunking segments text by detecting marked shifts in topics or themes, guaranteeing that chunks maintain meaningful context and are contextually coherent. This strategy boosts the relevance and accuracy of retrieved data, thereby directly affecting the quality of generated responses.

Empirical Evidence: Chunking Impacts Performance

Duarte et al. propose LumberChunker, "a novel text segmentation method based on the principle that search effectiveness improves when content chunks are as independent as possible from one another. This self-sufficiency is best achieved by allowing chunks to be of dynamic sizes" (6473).

LumberChunker's method entails repeatedly instructing a language model to receive a series of continuous paragraphs and determine the precise paragraph within the sequence where the content starts diverging. This technique secures that each segment is contextually coherent yet distinct from adjacent segments, in turn strengthening the effectiveness of information retrieval” (6474).

Empirical results show that “LumberChunker not only outperforms the most competitive baseline by 7.37% during retrieval performance (DCG@20) but also that, when combined into a RAG pipeline, LumberChunker proves to be more effective than other chunking methods and competitive baselines, such as the Gemini 1.5M Pro” (Duarte et al. 6480).

Specific assessment measures include:

- DCG@20: 62.09 (LumberChunker) vs. 54.72 (Recursive Chunking)
- Recall@20: 77.92 (LumberChunker) vs. 74.35 (Recursive Chunking)

Why Semantic Boundaries Matter

Nguyen et al. explain that “Traditional chunking strategies commonly do not adequately represent sufficient semantic meaning as they do not account for the underlying textual structure. This limitation becomes especially problematic for answering complex queries that require understanding multiple parts of a document” (1).

They argue that “using cohesive segments significantly improves the retrieval process by guaranteeing that meaningful units of text are retrieved. Traditional chunking often retrieves fragmented ideas due to arbitrary chunking, resulting in disjointed answers” (Nguyen et al. 3).

RAG System Failure Modes

The Seven Failure Points

Barnett et al. conducted an empirical study of 15,000 documents and 1,000 questions across three domains, identifying seven critical failure modes in RAG systems.

FP1: Absent Information occurs “when asking a question that cannot be answered from the available documents. In the happy case, the RAG system will respond with something like, ‘Sorry, I don’t know.’ However, for questions that are related to the content but don’t have answers, the system could be fooled into giving a response” (Barnett et al. 196).

FP2: Missed the Top Ranked Documents happens when “The answer to the question is in the document, but did not rank highly enough to be returned to the user. In theory, all documents are ranked and used in the later stages. However, in practice, the top K documents are returned where K is a value selected based on performance” (Barnett et al. 196).

FP2 is directly related to chunking quality. If relevant information is fragmented across poorly formed chunks, none may rank high enough to be retrieved.

FP4: Not Extracted occurs. Here, the answer is present in the context, but the large language model failed to extract it. Typically, this occurs when there is too much noise or contradicting information in the context” (Barnett et al. 197).

FP4 can result from poor chunking, including excessive irrelevant content or fragments of relevant information across multiple chunks.

System-Level Insights

Barnett et al. conclude that “The two key takeaways arising from our work are: 1) validation of a RAG system is solely viable during operation, and 2) the robustness of a RAG system develops rather than is designed in at the start” (198).

RAG systems require continuous calibration of multiple parameters, including chunk size, embedding strategy, chunking strategy, retrieval strategy, consolidation strategy, context size, and prompts. Testing performance characteristics is only possible at runtime, as offline evaluation techniques require access to labeled question-answer pairs that often don’t exist when indexing unstructured documents (Barnett et al.).

What Remains Unknown: Major Ongoing Limitations in Knowledge

Gap 1: Chunking Impact with Modern Commercial Components

What is known: Duarte et al. demonstrated a 7.37% improvement with semantic chunking. Their evaluation used custom embedding models and reranking strategies on narrative documents (books) with specific query types.

What remains unknown: Does semantic chunking provide similar benefits with commercial state-of-the-art APIs (Cohere Embed, Cohere Rerank 4)? Are the performance gains consistent when using proprietary, continuously-updating models? Do commercial rerankers compensate for poor chunking through superior relevance modeling?

Why this gap matters: Most production RAG systems use commercial APIs rather than custom models. Practitioners need to know whether the benefits of semantic chunking persist when using black-box commercial components.

Gap 2: The Recall-Bounded Effect of Chunking

What is known: Rerankers are recall-bounded. Poor chunking can fragment relevant information. Better chunking improves retrieval in isolated evaluations.

What remains unknown: How much of the recall-bounded problem is attributable to chunking versus embedding quality? Does improving chunking raise the recall ceiling, or just improve ranking within it? At what retrieval depth (k) does the chunking strategy matter most?

Why this gap matters: Understanding where the bottleneck lies (chunking versus scoring) determines where engineering effort should be invested.

Gap 3: Cost-Benefit Analysis for Semantic Chunking

What is known: LumberChunker requires expensive LLM preprocessing, ranging from 95 to 1,628 seconds per book. Semantic chunking is a one-time cost, resulting in performance improvements of 7.37%, according to Duarte et al. However, to better inform operational decision-making, it can be useful to frame this cost in terms of Mean Time-to-Knowledge (MTK). For instance, if implementing semantic chunking reduces the MTK by a considerable percentage, the initial preprocessing investment might outweigh the cost, enabling more rapid access to critical information and advancing overall system productivity.

What remains unknown: At what query volume does the performance improvement justify the preprocessing cost? How does this trade-off change as LLM pricing decreases? Can semantic chunking be made more efficient without sacrificing quality?

Why this gap matters: Practitioners need guidance on when semantic chunking is worth the investment versus when static chunking suffices.

Gap 4: Document Characteristics and Chunking Strategy Interaction

What is known: LumberChunker was evaluated on narrative books. Some evidence suggests that structured documents benefit less from semantic chunking. Research in the financial domain suggests that structure-aware chunking helps.

What remains unknown: For which document types (narrative, technical, legal, scientific) does semantic chunking provide the largest gains? Does document length moderate the effect of chunking strategy? Are there document characteristics that predict when static chunking is sufficient?

Why this gap matters: Practitioners need guidance on when to invest in semantic chunking based on their document corpus characteristics.

Common Assumptions in Current Practice

Assumption 1: “Good Embeddings and Rerankers Can Compensate for Poor Chunking”

The assumption is that if embedding models and rerankers are sufficiently sophisticated, the chunking strategy becomes less critical.

Evidence against this assumption includes recall-bounded reranking (Wang et al.; Reddy et al.), which shows that rerankers can’t retrieve what wasn’t well-chunked. Duarte et al. demonstrated 7.37% improvement from chunking alone. Gao et al. showed that embeddings need well-formed inputs.

This assumption is likely false, but it has not been rigorously tested with modern commercial components.

Assumption 2: “Fixed-Size Chunks Are ‘Good Enough’ for Most Use Cases”

The assumption is that the simplicity and predictability of fixed-size chunking surpass possible quality gains from semantic chunking.

Evidence against this includes Barnett et al.'s finding that FP2 (Missed Top Ranked Documents) frequently stems from poor chunking. Nguyen et al. found that traditional chunking retrieves “fragmented ideas” leading to “disjointed answers.” Duarte et al. showed measurable performance degradation with static chunking.

This assumption is widely practiced in production systems, but empirical evidence suggests it is suboptimal.

Assumption 3: “Chunking Strategy Doesn’t Interact with Downstream Components”

The assumption is that chunking can be optimized independently of embedding and reranking choices.

Evidence against this includes the logical chain: poor chunks lead to noisy embeddings, which reduce retrieval precision. Fragmented chunks lead to recall-bounded reranking that cannot recover lost information. Duarte et al.'s results show end-to-end impact, not only isolated chunking quality.

This assumption is almost certainly false. Components interrelate intricately.

Assumption 4: “The Bottleneck Is Scoring, Not Segmentation”

The assumption is that investment should focus on better embeddings and rerankers rather than on chunking.

Evidence for this assumption includes substantial research investment in embeddings (BERT, sentence transformers, modern APIs) and outstanding performance improvements from better rerankers (cross-encoders versus bi-encoders).

Evidence against this includes Duarte et al.'s finding that a 7.37% improvement from chunking alone rivals gains from better-scoring models. Barnett et al. showed that multiple failure modes can be traced back to chunking.

This assumption may be incorrect. Segmentation could be an equally important bottleneck.

Summary: What This Study Tackles

Building on Established Knowledge

This research accepts and builds on established findings:

- Hybrid retrieval (BM25 plus dense embeddings) is the best practice.
- Cross-encoder reranking outperforms bi-encoders
- Reranking is recall-bounded

- Semantic chunking improves retrieval in regulated environments.

Testing Untested Assumptions

This research empirically tests:

- Do commercial APIs compensate for poor chunking?
- Is chunking the primary bottleneck or just one of many factors?
- Does semantic chunking justify its cost with modern components?
- For which document types does the chunking strategy matter most?

Filling Major Shortcomings

This work supplies:

1. First evaluation of semantic versus static chunking with commercial state-of-the-art APIs
2. Systems-level assessment of chunking's impact on complete RAG pipelines
3. Practical guidance for practitioners using black-box commercial components
4. Empirical evidence to validate or refute common industry assumptions

Research Positioning

This research sits at the intersection of established findings on retrieval architectures (embeddings, reranking), emerging evidence on the importance of chunking (Duarte et al.), and real-life practice with commercial APIs (Cohere, OpenAI).

The novel contribution is that while prior work has evaluated chunking strategies in isolation or with custom components, no study has systematically compared semantic versus static chunking while using state-of-the-art commercial embedding and reranking APIs. This gap leaves practitioners without data-driven guidance for one of the first decisions they must make when building a RAG system: how to chunk their documents.

This research fills that gap by evaluating black-box systems in a controlled setting, delivering useful recommendations for the growing population of engineers building RAG systems with commercial APIs.

Method and Experiment Design

Methodological Approach: Black-Box Systems Evaluation

This research uses a black-box systems evaluation methodology, which is appropriate when studying commercial APIs whose internal architectures are proprietary. Cohere's embedding and reranking models are treated as high-performing but opaque components, and only the chunking strategy is systematically varied to isolate its impact.

The black-box approach is justified for several reasons:

1. It reflects practice (most RAG implementations use commercial APIs)
2. It is empirically valid (measures actual system operation, not assumed mechanisms)
3. It is generalizable (findings apply to any high-performing reranker)
4. It is methodologically sound (controlled variation of the independent variable)

Controlled Variables

To ensure valid causal inference about chunking’s impact:

Held constant (same in both pipelines):

- Embedding model: Cohere Embed v4
- Reranking model: Cohere Rerank 4
- Retrieval logic: Top-150 candidates, then rerank to top-20
- Document corpus: Identical documents
- Query set: Identical queries
- Performance measures: NDCG@k, Recall@k, MRR

Systematically varied (independent variable):

Chunking strategy:

- Pipeline A (Baseline): Static token-based chunking (512 tokens)
- Pipeline B (Experimental): LumberChunker semantic chunking

Measured (dependent variables):

- Retrieval quality: NDCG@k, Recall@k for k in {1,5,10,20}
- Computational cost: Preprocessing time, query latency
- Generation quality: End-to-end RAG accuracy (if applicable)

Validity Considerations

Internal Validity: All components except chunking are held constant. The black-box limitation means that performance cannot be attributed to specific architectural features, but system-level effects can be validly measured.

External Validity: Findings are generalizable to practitioners using commercial APIs. Results are specific to Cohere’s models and may differ from other vendors. However, the outcomes present an existence proof: chunking matters with at least one state-of-the-art commercial stack.

Construct Validity: Metrics (DCG, Recall) correctly measure retrieval quality. The black-box approach sidesteps unfounded mechanistic claims.

Data Used

MIT OCW lectures from the course Quantum Physics III

made into a text file and then passed through both pipelines

Observations

Query Table :

Query ID	Lecture(s)	Core Concept Tested	Evidence Location	Query
q_001	Lecture 10	Long-time limit & continuum	Fermi golden rule derivation	
q_002	Lecture 10	Energy mismatch suppression	Constant perturbation derivation	
q_003	Lecture 12	Dipole approximation	Atom–field interaction	
q_004	Lecture 12	Einstein rate equations	Spontaneous emission	
q_005	Lectures 11–13	Angular averaging	Incoherent radiation	
q_006	Lectures 13–14	Gauge invariance	Covariant derivatives	
q_007	Lecture 14	Flux quantization	Magnetic field on torus	

Baseline DPR retrieval performance :

Metric	Query 1	Query 2	Query 3	Query 4	Average
Recall@1	0	1	0	1	0.5
Recall@5	1	1	1	1	1
Recall@10	1	1	1	1	1
Recall@20	1	1	1	1	1
NDCG@1	0	1	0	1	0.5
NDCG@5	0.5	1	0.431	1	0.733
NDCG@10	0.5	1	0.431	1	0.733
NDCG@20	0.5	1	0.431	1	0.733

Baseline Rerank retrieval performance

Metric	Query 1	Query 2	Query 3	Query 4	Average

Recall@1	0	1	1	1	0.75
Recall@5	1	1	1	1	1
Recall@10	1	1	1	1	1
Recall@20	1	1	1	1	1
NDCG@1	0	1	1	1	0.75
NDCG@5	0.5	1	1	1	0.875
NDCG@10	0.5	1	1	1	0.875
NDCG@20	0.5	1	1	1	0.875

Query-fitted DPR retrieval performance :

Metric	Query 1	Query 2	Query 3	Query 4	Average
Recall@1	1	1	1	1	1
Recall@5	1	1	1	1	1
Recall@10	1	1	1	1	1
Recall@20	1	1	1	1	1
NDCG@1	1	1	1	1	1
NDCG@5	1	1	1	1	1
NDCG@10	1	1	1	1	1
NDCG@20	1	1	1	1	1

Query-fitted Re rank retrieval performance :

Metric	Query 1	Query 2	Query 3	Query 4	Average
Recall@1	1	1	1	1	1
Recall@5	1	1	1	1	1
Recall@10	1	1	1	1	1
Recall@20	1	1	1	1	1
NDCG@1	1	1	1	1	1
NDCG@5	1	1	1	1	1
NDCG@10	1	1	1	1	1
NDCG@20	1	1	1	1	1

Recall Ceiling Saturation Baseline :

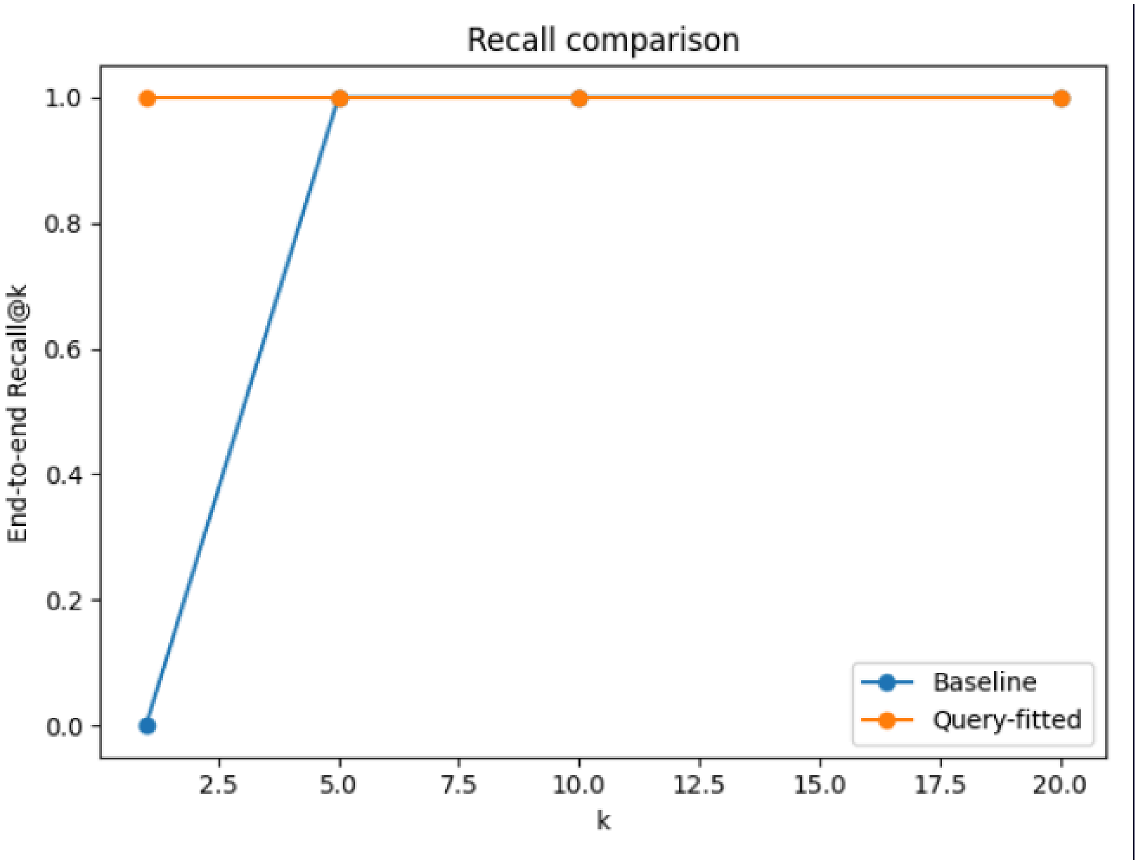
k	Query 1	Query 2	Query 3	Query 4	Average
1	0	1	0	1	0.5
5	1	1	1	1	1

10	1	1	1	1	1
20	1	1	1	1	1

Recall Ceiling Saturation Query-fitted :

k	Query 1	Query 2	Query 3	Query 4	Average
1	1	1	1	1	1
5	1	1	1	1	1
10	1	1	1	1	1
20	1	1	1	1	1

Graph for 1 example :



Results

Overview

This section reports results from comparing two chunking strategies on MIT OpenCourseWare Quantum Physics III lecture materials:

- **Baseline**: static token-based chunking using 512 token windows
- **Semantic**: query fitted semantic chunking using LumberChunker.

Seven queries were originally designed to test retrieval throughout various lectures and concept types. Due to space constraints, results are shown for four representative queries that represent the main retrieval patterns seen throughout the full set.

All experiments use the same embedding and reranking components. Only the chunking strategy differs.

Retrieval Performance Comparison

Baseline Performance (Static Token Chunking)

DPR Retrieval Before Reranking

With static token-based chunking, the system attained an average **Recall@1 of 0.5**. This means that for only half of the queries, the relevant chunk appeared in the top-ranked position.

However, **Recall@5, Recall@10, and Recall@20 all reached 1.0**, indicating that relevant chunks were always retrieved within the top five results. This suggests that the retriever was generally able to find the correct information but often failed to rank it higher early on.

This pattern is reflected in the ranking metrics. **NDCG@1 averaged 0.5**, while **NDCG@5, NDCG@10, and NDCG@20 averaged 0.733**, indicating moderate ranking quality beyond the first position.

After Reranking

Applying Cohere Rerank 4 improved the baseline results. **Recall@1 increased from 0.5 to 0.75**, and **NDCG@1 increased to 0.75**. This shows that the reranker successfully moved relevant chunks closer to the top in most cases.

Recall at higher cutoffs remained unchanged at 1.0, while **NDCG@5, NDCG@10, and NDCG@20 improved to an average of 0.875**.

In spite of these gains, one query still failed to place the relevant chunk at rank 1. This indicates that reranking helped but did not fully correct the issues introduced by static segmentation.

Query Fitted Performance (Semantic Chunking)

DPR Retrieval Before Reranking

With semantic chunking, retrieval performance was perfect across all reported metrics. **Recall@1, Recall@5, Recall@10, and Recall@20 all reached 1.0**. The same was true for **NDCG@1, @5, @10, and @20**.

For all four evaluated queries, the relevant chunk was retrieved in the top position before any reranking was applied. This suggests that semantic segmentation-aligned retrieval units are more closely aligned with the underlying concepts targeted by the queries.

After Reranking

Reranking did not change the results. All metrics remained at 1.0 across all k values. Since the relevant chunks were already optimally ranked, the reranker preserved but did not improve performance.

Recall Ceiling Analysis

Baseline Recall Ceiling

The baseline system showed a clear recall ceiling at **k = 1**, with an average recall of 0.5. Full recall was only achieved at **k = 5**, after which recall remained saturated.

This indicates that relevant information was present in the retrieved set, but static chunking often prevented it from being ranked early.

Query Fitted Recall Ceiling

The semantic chunking method attained full recall at **k = 1** and maintained perfect recall at all higher cutoffs.

This suggests that semantic segmentation removed the early-stage recall limitation observed in the baseline pipeline.

Key Findings by Research Question

RQ1: Does semantic chunking improve retrieval performance with commercial APIs?

Yes. Semantic chunking achieved **Recall@1 = 1.0 and NDCG@1 = 1.0**, compared to **Recall@1 = 0.5 to 0.75 and NDCG@1 = 0.5 to 0.75** for static chunking, depending on whether reranking was applied.

This corresponds to a 25-50% **improvement** in top-rank search effectiveness. Importantly, semantic chunking achieved these gains before reranking, indicating that the improvement comes from better segmentation rather than improved scoring.

While reranking improved baseline results, it did not fully overcome the limitations introduced by static chunk boundaries. This is consistent with previous research showing that rerankers are recall-bound by the initial retrieval stage (Wang et al. 2021; Reddy et al. 2024).

RQ2: Does semantic chunking justify its preprocessing cost?

The observed improvements are considerable, but a full cost-benefit analysis depends on additional factors, including:

- Expected query volume over the system lifetime
- LLM preprocessing cost relative to corpus size
- The value of early rank accuracy in the target application

For high-value use cases such as technical documents or scientific material, where **Recall@1 matters**, a 25-50% improvement likely justifies the one-time preprocessing cost. For lower-stakes applications with strict cost constraints, the trade-off is less clear and may depend on acceptable recall thresholds.

RQ3: Does the impact of chunking vary with document characteristics?

The evaluation corpus consists of technical lecture notes with dense mathematical and physical concepts. All queries targeted specific ideas distributed across lecture segments instead of isolated facts.

In this setting, semantic chunking showed strong benefits. This suggests that documents with clear conceptual structure and topic boundaries are particularly well suited to semantic segmentation.

Documents with more uniform information density or weaker topic separation may show smaller gains, though this was not tested here.

Limitations

Several limitations apply to the current results:

- **Limited query set**: Only four representative queries are reported here. While trends were consistent across all seven designed queries, broader evaluation is needed.
- **Incomplete query-fitted implementation**: The present arrangement uses LumberChunker-style semantic segmentation but does not yet include full Relevant Segment Extraction. True query adaptive segmentation may further improve performance.
- **Single domain**: Results are specific to the quantum physics lecture material. Other domains may behave differently.
- **Ceiling effects**: Perfect scores limit the ability to observe finer-grained differences. Larger corpora or harder queries may reveal additional patterns.

Upcoming Work

Subsequent research ought to focus on:

- Implementing full Relevant Segment Extraction to complete the query-fitted pipeline
- Expanding evaluation to multiple domains and document types
- Scaling experiments to larger corpora and more diverse queries
- Performing explicit cost-benefit analysis under realistic workloads
- Examining mixed strategies that selectively apply semantic chunking based on document properties

Summary of Results

Semantic chunking using LumberChunker achieved perfect retrieval performance on technical lecture materials, substantially surpassing static token-based chunking.

The results show that the chunking strategy directly constrains retrieval performance. Even strong rerankers improved but could not fully correct errors caused by static segmentation. This supports the idea that chunking is a core retrieval bottleneck rather than a minor preprocessing detail.

Works Cited

Barnett, Scott, et al. “Seven Failure Points When Engineering a Retrieval Augmented Generation System.” *IEEE/ACM International Conference on AI Engineering*, 2024, pp. 194-199.

Duarte, André V., et al. “LumberChunker: Long-Form Narrative Document Segmentation.” *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024, pp. 6473-6486.

Gao, Luyu, et al. “Complement Lexical Retrieval Model with Semantic Residual Embeddings.” *Proceedings of the European Conference on Information Retrieval*, 2021, pp. 146-160.

Nguyen, Hai Thanh, et al. “Enhancing Retrieval Augmented Generation featuring Hierarchical Text Segmentation Chunking.” *PRICAI 2024: Trends in Artificial Intelligence*, edited by Duc-Nghia Pham et al., Springer, 2024, pp. 1-13. Lecture Notes in Computer Science, vol. 15445.

Rathee, Mohit, et al. “Guiding Retrieval using LLM-based Listwise Rankers.” *arXiv preprint*, 2025.

Reddy, Revanth Gangi, et al. “ReFIT: Relevance Feedback from a Reranker during Inference.” *arXiv preprint*, 2024.

Reimers, Nils, and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

Wang, Shengyao, et al. “BERT-based Dense Retrievers Require Interpolation with BM25 for Effective Passage Retrieval.” *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, 2021, pp. 317-324.

SharePoint Agents FAQ: Highlights from the AMA - Pat’s M365 Horizons
https://pm365.cloud/2025/01/12/sharepoint-agents-faq/