Improving Retrieval-Augmented Generation for Financial Document Question Answering via Multi-Stage Retrieval and Reranking

Kent Goodman, Jeffrey Li, Kaylee Sweet, Nathan Zhu
MIT
{kentg, jefffrey, kaysweet, nathannn}@mit.edu

Abstract

Financial question answering over corporate filings remains a core challenge for large language models (LLMs). Modern filings routinely span hundreds of pages, contain dense accounting and regulatory language, and—especially in private markets—are not indexed by web search, preventing LLMs from supplementing missing knowledge externally. Naively providing full documents to an LLM is infeasible: a single query across multiple filings in our evaluation consumed more than 800,000 input tokens, making raw LLM usage prohibitively expensive and often exceeding model context limits. Even standard Retrieval-Augmented Generation (RAG) pipelines struggle when documents exhibit high structural similarity, as shallow embedding-based retrieval returns large sets of near-duplicate chunks that overwhelm the model with noise rather than evidence.

We present a multi-stage RAG architecture designed to operate under the cost, context, and relevance constraints of real-world financial analysis. Our system combines (i) a MarkItDown preprocessing stage to structure documents and route queries to relevant sections, (ii) dense retrieval to select candidate passages, (iii) a cross-encoder reranker optimized for identifying answer-bearing evidence, and (iv) a reduced-context reasoning stage that forces the LLM to operate only on the most informative passages. This workflow enables accurate reasoning across long filings without exceeding context budgets or incurring extreme token costs.

Evaluated on a randomly sampled set of human-written financial analysis questions, our method consistently outperforms both baseline RAG and raw LLM generation. Using an LLM-as-judge framework, we find that the workflow improves correctness from **72% to 92%** (a 20-point gain), achieves an 81% head-to-head preference rate over baseline RAG, and reduces token usage by an order of magnitude compared to raw GPT processing. In cost terms, our workflow is over **100× cheaper** than raw LLM querying of full filings. These results demonstrate that retrieval architecture—rather than model size or prompting alone—is the primary driver of reliable and cost-efficient financial QA at scale.

1 Introduction

Large language models (LLMs) have achieved strong performance on open-domain question answering, summarization, and general reasoning. However, their effectiveness drops sharply when applied to long, domain-specific financial documents such as annual 10-K filings. These filings often exceed hundreds of pages, contain heterogeneous disclosures across accounting, operations, and regulatory sections, and require nuanced cross-referencing to answer even seemingly simple questions. In private markets, the challenge is compounded: financial documents are proprietary and absent from web search indices, preventing LLMs from retrieving external context.

A central yet underexplored obstacle in this setting is the *interaction between document length, LLM context limits, and inference cost*. Modern long-context models still fall far short of what real financial workflows require. For example, GPT-5-mini supports a maximum of 250,000 input tokens—far below the size of even a small collection of filings. When multiple annual reports are provided in full, token consumption becomes extreme. In our empirical analysis, a single question spanning five filings required over 800,000 input tokens, immediately exceeding the model's context window and making such queries impossible for smaller or cost-optimized LLMs.

Even for models that technically support these larger contexts, cost becomes prohibitive. GPT-4.1 pricing is approximately $3 per million input tokens, implying that 100 such queries would exceed $300 in inference cost. This level of expense renders direct LLM usage infeasible for any production system aimed at automating financial document analysis, due diligence, or large-scale corporate research. Without aggressive retrieval, filtering, and context reduction, LLMs cannot be used economically or reliably in real-world multi-document financial workflows.

Retrieval-Augmented Generation (RAG) is intended to mitigate these issues by providing the model with only the most relevant fragments of text. Yet standard RAG pipelines break down in financial settings. Filings across companies and years share highly similar structure and vocabulary, causing shallow embedding-based retrieval to surface large sets of near-duplicate chunks. These distractors pollute the model's context, degrade grounding, and frequently lead to hallucinations or misinterpretations. In high-stakes environments—where a single incorrect figure can materially affect valuation or risk assessments—these failure modes are unacceptable. Several recent real-world incidents illustrate how LLM hallucinations in due diligence can lead to multi-million-dollar consequences.

To address these limitations, we propose a multi-stage RAG architecture designed specifically for large-scale financial document question answering. The system begins with a dedicated *document processing* stage that converts raw corporate filings into structured, searchable units. Filings—often hundreds of pages in PDF format—are parsed, segmented, and normalized into consistent markdown sections, creating stable chunk boundaries and preserving the structural cues that are essential in financial documents.

Building on this processed representation, the system performs retrieval and filtering through the following components:

1. **Document-level selection**: a routing stage identifies which filings are likely to contain relevant information, preventing retrieval across hundreds of pages of irrelevant material and ensuring that only the most promising documents enter the finer-grained retrieval pipeline.

2. **Document processing**: raw PDFs are parsed, cleaned, and segmented into structured text with consistent headings and chunk boundaries, enabling efficient downstream retrieval.

3. **Chunk-level embedding search**: within the selected documents, embedding-based similarity retrieval produces an initial set of candidate passages to select relevant candidates for reranking.

4. **Cross-encoder reranking**: a fine-grained relevance model reevaluates all candidate passages in context with the query, sharply discriminating between answer-bearing evidence and structurally similar distractors common in financial filings.

5. **Reduced-context querying**: the LLM receives only the highest-value passages from the reranking stage, enabling grounded reasoning while keeping token usage—and therefore inference cost—extremely low.

This hierarchical retrieval flow enables accurate multi-document financial analysis while remaining within strict context and cost constraints, addressing the primary failure modes of raw LLMs and standard RAG pipelines. In our evaluation on human-written financial analysis questions, the system consistently outperforms both baseline RAG and raw LLM generation across correctness, evidence grounding, and hallucination reduction.

These results highlight a central insight of this work: *retrieval architecture—not model size or prompting strategy—is the critical bottleneck for accurate LLM-based financial QA*. The proposed approach offers a practical and scalable blueprint for deploying LLM systems in real-world financial environments where document volume, privacy limitations, and accuracy requirements exceed the capabilities of standard methods.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) has emerged as a primary approach for extending large language models beyond their fixed pretraining corpora by retrieving relevant external documents at inference time. Early RAG systems typically rely on dense embedding retrieval followed by generation over the retrieved context, demonstrating strong performance on open-domain question answering tasks with relatively short and heterogeneous documents [Lewis et al., 2020]. However, subsequent work has shown that naive RAG pipelines degrade substantially as document length increases, both due to retrieval noise and context window constraints [Guu et al., 2020, Izacard and Grave, 2021].

Several studies have proposed adaptations of RAG for domain-specific settings. Wang et al. Wang et al. [2025a] incorporate finance-oriented preprocessing and retrieval to improve question answering accuracy on financial texts, showing that domain-aware chunking and vocabulary alignment can outperform generic retrieval pipelines. Nevertheless, their evaluation focuses on relatively short passages and does not address the challenges posed by full-length corporate filings, which routinely exceed hundreds of pages and contain heterogeneous content such as narrative disclosures, tables, and cross-referenced footnotes. As a result, the scalability of these approaches to realistic analyst workflows remains unclear.

More generally, prior work highlights that RAG performance is often limited not by generation quality but by retrieval failures, specifically when relevant evidence is distributed across multiple document sections. This limitation is amplified in financial filings, where repeated structural patterns (e.g., standardized risk factor language) lead embedding-based retrievers to return large sets of near-duplicate passages. Existing RAG benchmarks and architectures rarely evaluate this failure mode explicitly.

2.2 Semantic Reranking

To mitigate the limitations of embedding-only retrieval, recent work has explored semantic reranking using cross-encoders or late-interaction models. These approaches apply a more expressive relevance model to a small candidate set produced by a high-recall retriever, substantially improving precision for downstream reasoning [Nogueira and Cho, 2019, Khattab and Zaharia, 2020]. Cross-encoder rerankers have proven especially effective in identifying answer-bearing passages in settings where lexical or embedding similarity alone is insufficient.

Multi-stage retrieval pipelines that combine retrieval, reranking, and iterative reasoning have shown strong gains in general-domain multi-hop QA. Tang and Yang [2024] introduce MultiHop-RAG to evaluate systems that must retrieve and integrate evidence across multiple reasoning steps, revealing that even state-of-the-art retrievers struggle to assemble complementary passages. Liu et al. [2025]'s further improve multi-hop retrieval by constructing a passage graph and iteratively pruning irrelevant nodes, demonstrating that retrieval architecture is a primary bottleneck in complex QA tasks.

Despite their success, these reranking and multi-hop systems are largely evaluated on Wikipedia-style corpora with relatively uniform structure. Financial documents pose distinct challenges: relevant evidence may span narrative explanations, numerical tables, and footnotes that are semantically related but weakly connected in embedding space. Moreover, few prior works explicitly optimize reranking models for identifying answer-bearing financial evidence rather than generic relevance. An exception is FinGEAR [Wang et al., 2025b], which introduces a finance-specific retrieval and cross-encoder reranking pipeline tailored to long regulatory filings, demonstrating that domain-aware reranking can substantially improve evidence selection in financial question answering.

2.3 Financial QA and Document Understanding

Financial question answering has attracted increasing attention due to the importance of accurate and explainable analysis in regulatory and investment contexts. Prior work has explored domain adaptation, specialized pre-training, and structure-aware processing to address the mismatch between general-purpose LLMs and financial language [Araci, 2019]. Yepes et al. [2024] demonstrate that chunking long financial documents according to detected structure, such as section headers and bullet groups, significantly improves retrieval effectiveness compared to sliding-window approaches. However, their method does not address multi-hop reasoning across distant sections, limiting its applicability to complex analytical questions.

Several benchmarks and datasets have been proposed for financial document understanding, including question answering over earnings reports and regulatory filings. Yet many evaluations risk data leakage: training and test splits often include filings from the same companies or adjacent time periods, allowing models to exploit recurring structure rather than genuine reasoning. This concern is consistent with findings from LastingBench [Li et al., 2024], which shows that long-context QA benchmarks frequently overestimate model performance due to memorization and knowledge leakage rather than true document-grounded reasoning. In addition, few works require explicit attribution or enforce grounding of answers in cited passages, despite the central role of traceability and trustworthiness in financial analysis.

Overall, existing research has examined retrieval strategies, semantic reranking, and financial QA largely in isolation. To our knowledge, no prior system integrates structure-aware chunking, multi-stage retrieval with semantic reranking, and strict evidence grounding into a single pipeline designed for long, highly repetitive financial filings. Our work addresses this gap by demonstrating that retrieval architecture, rather than model size or prompting alone, is the key driver of reliable and cost-efficient financial question answering at scale.

3 Methodology

In this section, we describe the end-to-end architecture of our financial QA workflow. The system takes a set of raw PDF filings, preprocesses them into structured text, routes each query to a small subset of relevant documents, performs dense retrieval over chunked text, optionally reranks retrieved passages, and finally queries an LLM with a reduced context window. We denote the collection of input documents by $\mathcal{D} = \{D_1, \ldots, D_N\}$ and a user question by q.

Figure 1 provides a high-level overview of the pipeline, and the following subsections detail each component.

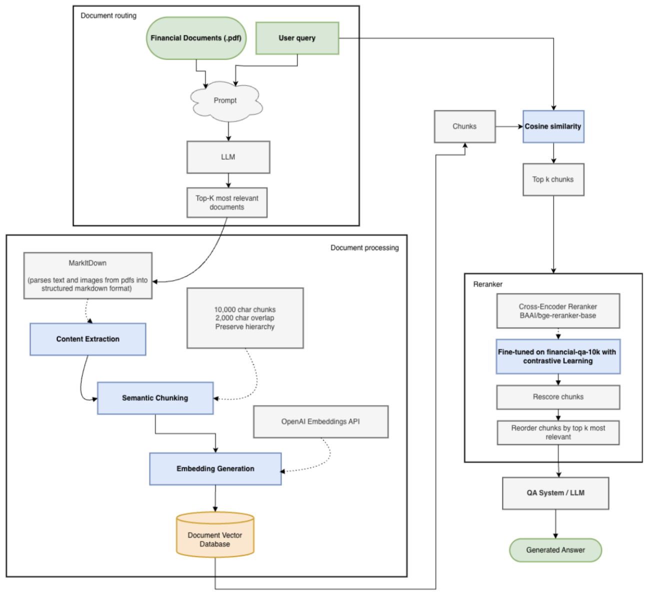

Figure 1: Multi-stage retrieval and reranking model pipeline

3.1 Problem Setting

Given a collection of corporate filings $\mathcal{D}$ and a natural-language query q, our goal is to generate an answer a that is (i) factually grounded in the contents of $\mathcal{D}$ and (ii) obtained within realistic LLM context and cost constraints. Formally, we seek:

$$a = \mathrm{LLM}(q, \mathcal{C}(q, \mathcal{D})),$$

where $\mathcal{C}(q, \mathcal{D})$ is a small set of context passages selected from $\mathcal{D}$ by our retrieval pipeline. The core challenge is to construct $\mathcal{C}$ so that it contains answer-bearing evidence while remaining orders of magnitude smaller than the full corpus.

3.2 Document Processing and Chunking

PDF to structured text. We first convert each raw PDF filing D_i into a normalized, searchable representation T_i. We use a custom processor based on MarkItDown, which:

- parses the PDF,
- emits markdown-like structured text with headings, sections, and tables preserved, and
- integrates our internal image-query system to attach text descriptions for figures when needed.

This step produces a long text string per document while retaining much of the original financial structure (e.g., section headers, risk factor blocks), which is important for downstream retrieval.

Dual-scale chunking. Each processed document T_i is then split into overlapping chunks at two granularities:

- **Normal chunks:** length $L = 10{,}000$ characters with overlap $O = 2{,}000$ characters.
- **Reduced chunks:** length L_{red} and overlap O_{red} (500 and 250 characters in our experiments).

Chunking is implemented via a sliding window:

$$\text{chunk}_j = T_i[s_j : s_j + L], \quad s_{j+1} = s_j + (L - O),$$

and analogously for the reduced setting. The large chunks are designed to capture broad context with enough overlap to avoid boundary issues (any local evidence remains in at least one chunk), while the reduced chunks provide a more fine-grained index for lower-cost experiments and ablations. Both versions are stored in a simple in-memory database:

$$\mathcal{C}^{\text{normal}} = \{(c, \text{source}(c))\}, \quad \mathcal{C}^{\text{reduced}} = \{(c', \text{source}(c'))\},$$

where $\text{source}(\cdot)$ records the originating document path.

3.3 Metadata Extraction

To support document-level routing, we extract lightweight metadata for each filing. For each processed document T_i, we prompt a small LLM (GPT-4o-mini) with an excerpt of the first 3,000 characters and ask it to output a JSON object containing:

- company name (e.g., “NVIDIA Corporation”),

- fiscal year (e.g., “2023”),
- document type (e.g., “10-K”, “10-Q”, “Annual Report”), and
- a 2–3 sentence summary of key topics.

The workflow stores a metadata record

$$m_i = (\text{path}_i, \text{company}_i, \text{year}_i, \text{type}_i, \text{summary}_i)$$

for each document D_i. This metadata is used only for routing; we do not embed it or use it directly in chunk-level retrieval.

3.4 LLM-Based Document Routing

Given a query q and metadata $\{m_i\}_{i=1}^{N}$, we first restrict the search space to a small subset of likely-relevant filings. Rather than relying on cosine similarity over summary embeddings, we use an LLM to select documents via structured reasoning over company names, years, and document types.

We construct a prompt that lists the query and the available documents in indexed form:

1. Company: c_1 | Year: y_1 | Type: t_1Summary: s_1⋮

and instruct the model to choose the top-k most relevant documents. We enforce a JSON schema with fields: `selected_documents` (a list of indices) and `reasoning` (a brief explanation), and decode the response to obtain an index set $S(q) \subseteq \{1, \ldots, N\}$. The routed document paths are:

$$\mathcal{D}_{\text{route}}(q) = \{\text{path}_i : i \in S(q)\}.$$

In our experiments we typically use $k = 2$–3, motivated by the observation that many questions rely on one primary filing plus one or two additional filings that provide historical or comparative context.

If routing fails (e.g., due to API errors), we fall back to a simple heuristic that selects the first k documents, preserving robustness while keeping the primary evaluation focused on the LLM-based routing behavior.

3.5 Embedding-Based Chunk Retrieval

For the routed documents $\mathcal{D}_{\text{route}}(q)$, we create dense vector indexes via the `DocumentDatabaseParralized` class. For each chunk c_j associated with information type τ (e.g., “financial_documents”), we compute an embedding:

$$\mathbf{v}_j = f(c_j),$$

using OpenAI’s `text-embedding-3-large` model. Embeddings are computed in parallel using a thread pool to amortize API latency, and stored alongside their corresponding text and source paths.

At query time, we embed the query:

$$\mathbf{q} = f(q),$$

and compute cosine similarity between $\mathbf{q}$ and each chunk embedding $\mathbf{v}_j$:

$$\text{sim}(\mathbf{q}, \mathbf{v}_j) = \frac{\mathbf{q} \cdot \mathbf{v}_j}{\|\mathbf{q}\|_2 \, \|\mathbf{v}_j\|_2}.$$

We then select the top-K chunks by similarity:

$$\mathcal{C}_{\text{retr}}(q) = \text{TopK}_j \, \text{sim}(\mathbf{q}, \mathbf{v}_j),$$

optionally applying a similarity threshold. In the implementation, we typically retrieve $K = 20$ chunks (or $3K$ before filtering when routing is used), and we restrict the candidate set to chunks whose source path lies in $\mathcal{D}_{\text{route}}(q)$.

The retrieval can be performed over either the normal or reduced index:

- **Normal index** (`reduced=False`): larger chunks, higher per-chunk context, used for full evaluations.
- **Reduced index** (`reduced=True`): smaller chunks, lower token usage, used for cost-sensitive ablations and sanity checks.

3.6 Reranking: Current Heuristic and Future Work

Instead of applying a learned cross-encoder reranker, we rely on the similarity scores produced by the dense retriever and perform top-K truncation to get the final context set. Let

$$\mathcal{C}_{\text{retr}}(q) = \{(c_j, s_j)\}_{j=1}^{M}$$

denote the set of retrieved chunks and their cosine similarities s_j to the query embedding. We keep only the top K' chunks after sorting $\mathcal{C}_{\text{retr}}(q)$ in descending order by s_j:

$$\mathcal{C}_{\text{final}}(q) = \text{TopK}'_j \, s_j, \quad K' \ll M,$$

which are then concatenated and passed to the answer-generation model. All results reported in Section 5.1 and Section 5.2 use this simple top-K truncation scheme as the only reranking step.

This design isolates the effects of document routing and dense retrieval under realistic context and cost constraints, without conflating them with the behavior of a more expressive reranker. In future work, we plan to replace heuristic top-K truncation with a learned cross-encoder reranking model trained on financial QA evidence pairs. Such a model would score query–chunk pairs jointly and is expected to better discriminate between answer-bearing passages and structurally similar distractors when filings share highly repetitive language and table templates.

3.7 Answer Generation and Prompting

Given the final set of context chunks $\mathcal{C}_{\text{final}}(q)$, we construct a context string by concatenating the chunk texts separated by delimiters. Let $\{c_{j_1}, \dots, c_{j_{K'}}\}$ denote the selected chunks; we form:

$$\text{Context}(q) = c_{j_1} \parallel c_{j_2} \parallel \dots \parallel c_{j_{K'}},$$

where $\parallel$ denotes concatenation with simple separators (e.g., `"\n\n--\n\n"`).

We then query an LLM with a task-specific prompt drawn from `configs/prompts.json`. Our default configuration uses:

- a **system message**: “You are a financial analyst assistant. Answer questions based on financial documents.”
- a **user template** that injects both the question and the retrieved context, and explicitly instructs the model to answer *only* based on the provided documents and to say “I cannot find this information in the provided documents” if the answer is unavailable.

Formally, we call:

$$a = \text{LLM}\big(\text{system\_message},\ \text{user\_template}(q, \text{Context}(q))\big)$$

using `gpt-5-mini` as the main answering model to balance quality and cost. We log the number of context characters and API-reported token usage per query, enabling a direct comparison of cost between our system and naive full-document baselines.

3.8 End-to-End Workflow

Algorithmically, the `run()` method in our implementation executes:

1. Route the query q to a small set of documents $\mathcal{D}_{\text{route}}(q)$ using LLM-based routing over metadata;
2. Embed and index only the routed documents (if not already indexed);
3. Retrieve an initial set of chunks $\mathcal{C}_{\text{retr}}(q)$ via dense embedding search restricted to $\mathcal{D}_{\text{route}}(q)$;
4. Rerank (or truncate) to obtain $\mathcal{C}_{\text{final}}(q)$;
5. Query the LLM with $\mathcal{C}_{\text{final}}(q)$ as context to obtain the final answer.

This design directly encodes the cost- and context-aware principles discussed in Section 1, and forms the basis for the experimental comparisons in the next section.

4 Experimental Setup

4.1 Dataset

Our corpus consists of financial filings from **69 publicly traded companies**. For each company, we collect:

- the most recent 10-K filing,
- the prior year’s 10-K filing, and
- three additional 10-K filings from randomly selected companies serving as distractor documents.

This produces a realistic environment containing numerous structurally similar but semantically distinct long documents, mirroring real due-diligence and multi-firm analysis settings. All PDF filings are parsed using our MarkItDown-based processing pipeline to produce structured markdown with headings, tables, captions, and embedded image references preserved.

4.2 Question Set

Our full question corpus contains **6,900** human-written financial analysis questions. We use a set of **100 questions sampled uniformly at random** for our main evaluation, and all systems are evaluated on the exact same 100 sampled questions. These questions cover:

- factual financial lookups,
- multi-step reasoning,
- cross-document reconciliation,
- temporal comparisons across years,

- conceptual or risk-oriented disclosures.

All systems receive identical inputs and identical question order.

4.3 Document Sampling per Question

For the main end-to-end evaluation, we use a paired document-sampling protocol over questions and filings. From the full corpus of 69 companies, we first identify the target company and filing year associated with each question. For each of the 100 questions used in our primary comparison, we then randomly sample a subset of between 2 and 15 filings from the corpus. This subset always includes:

- the target company's current 10-K filing (the filing from which the question was derived),
- the target company's prior-year 10-K filing, and
- a random set of additional 10-K filings from other companies serving as distractors.

The resulting mixture of relevant and distractor filings mimics realistic due-diligence settings where analysts must reason over multiple structurally similar documents. The exact same sampled subset of filings is used for all systems being compared, ensuring that performance differences arise from the retrieval architecture rather than from document selection.

4.4 Models

All retrieval systems use `gpt-5-mini` for inference. However, **raw GPT baselines** required `gpt-4.1` due to context window limitations of `gpt-5-mini` ($\approx$ 238,000 tokens).

- Embedding-model: `text-embedding-3-large`
- Baseline RAG: similarity search over all chunks
- FinancialQA workflow: routing $\rightarrow$ search $\rightarrow$ rerank $\rightarrow$ LLM query
- Raw LLM (1-PDF and 5-PDF variants): `gpt-4.1`

4.5 Retrieval Configuration and Context Budget

Selecting chunk size and retrieval depth involves a trade-off between recall, context utilization, and inference cost. Larger chunks increase the probability of capturing relevant evidence but very quickly consume the model's context window and increase token costs, while smaller chunks require higher retrieval depth to maintain recall, reintroducing noise and reducing downstream effectiveness. Without an explicit constraint, choosing chunk sizes and top-k values leads to brittle or unrealistic configurations that fail to reflect real-world financial analysis workloads.

To ground this design space in a practical setting, we impose a hard constraint of 100,000 characters on the total context passed to the LLM, reflecting a realistic cost ceiling for repeated financial queries. Under this constraint, we evaluated multiple combinations of chunk sizes and retrieval depths, measuring retrieval recall relative to context consumption. We found that a chunk size of 10,000 characters with a top-k of 10 provides the best price vs. recall trade-off, consistently capturing relevant evidence while remaining well within the context budget after reranking. We adopt this configuration throughout our experiments as a cost-aware default.

4.6 Evaluation Metric

We use an LLM-as-judge framework to evaluate correctness. The judge model receives:

- the question,
- the generated answer,
- the ground truth answer,
- optional company context.

The judge applies strict binary evaluation: **correct** or **incorrect** only.

The evaluation prompt is shown below:

> Rules: - If the response includes the ground truth fact(s), mark it as CORRECT - Additional context is allowed - Mark INCORRECT only if the core fact is wrong or missing - Do not infer unstated information

We report:

- **Percent correct**
- **Pairwise preference** (Section 5.2.2)

No partial credit is assigned.

5 Results

5.1 Experiment 1: Feasibility, Token Costs, and Context Limits

A central challenge in financial QA is the tension between document length and LLM context limits. We therefore measure the feasibility and cost of three baseline strategies:

1. Raw GPT with full PDFs (1 PDF per query)
2. Raw GPT with all PDFs (5 PDFs per query)
3. Standard RAG (similarity search over all chunks)
4. Our FinancialQA workflow

5.1.1 Token Usage

Uploading a single 10-K to `gpt-4.1` for each query consumed on average:

167,000 tokens/query.

Attempting to upload all five filings per query produced:

835,000 tokens/query,

which exceeds the 238k context limit of `gpt-5-mini` and caused `gpt-4.1` to fail often but not always due to operational input size limits. The model successfully processed only 7 queries before encountering repeated context-overflow failures. All reported numbers for the 5-PDF raw GPT condition are therefore extrapolated from these seven successful runs.

By contrast:

Baseline RAG: $\approx$ **13,000 tokens/query**

Our workflow: $\approx$ **16,000 tokens/query**

5.1.2 Cost Comparison

Using OpenAI pricing:

- `gpt-4.1` input: $3.00 / 1M tokens
- `gpt-5-mini` input: $0.25 / 1M tokens

The resulting cost profile for 100 questions is shown in Table 1.

Method	Avg tokens/q	Cost (100q)	Feasible?
Raw GPT (1 PDF)	167k	$67	✓
Raw GPT (5 PDFs)	835k	$> \$300$	×
Baseline RAG	13k	$< \$1$	✓
Our workflow	16k	$< \$1$	✓

Table 1: Token usage and estimated cost for different methods on 100 queries. Raw GPT (5 PDFs) often exceeded model context limits; costs extrapolated from successful runs.

5.1.3 Discussion

Raw GPT is either:

- **infeasible** (context too small), or
- **economically prohibitive** ($67–$300 per 100 questions).

Standard RAG is inexpensive but suffers from severe accuracy degradation when many structurally similar filings are present.

Our workflow matches baseline RAG in cost while enabling robust retrieval under noise.

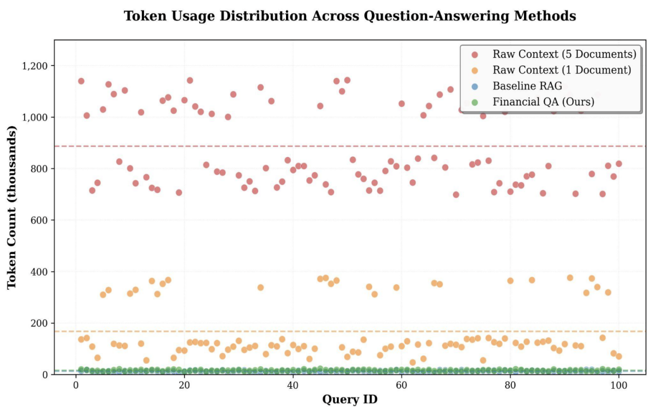

Figure 2: Token count comparison across QA methods

5.2 Experiment 2: End-to-End Accuracy on Financial QA Benchmark

We next evaluate answer correctness on the full financial QA benchmark.

5.2.1 Accuracy Results

To ensure a controlled and statistically sound comparison, we evaluate both systems on the **same 100 questions**, sampled uniformly at random from our full set of 6,900 human-written financial analysis questions.

Method	Accuracy (%)
Baseline RAG	72%
Our FinancialQA workflow	92%

Table 2: Answer correctness on the financial QA benchmark. Our workflow improves accuracy by 20 percentage points over baseline RAG while operating at comparable token cost (Table 1).

We use the document sampling procedure described in Section 4 for each question. The procedure selects 2–15 filings including the target company's 10-K, its prior-year 10-K, and then random distractor filings. The same sampled subset is used for both systems to ensure the evaluation is controlled and paired.

Table 2 reports the percentage of questions answered correctly according to the LLM-as-judge metric described in Section 4.

Our workflow achieves **92%** correctness compared to **72%** for baseline RAG, a **20 percentage point absolute gain**. Both systems use `gpt-5-mini` as the answer model, and both are evaluated on the same questions and ground-truth annotations, isolating the effect of retrieval architecture and prompt engineering rather than model capacity.

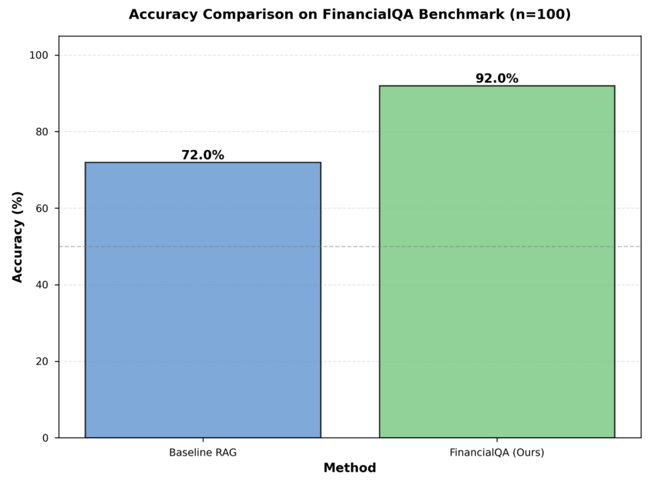

Figure 3: Token count comparison across QA methods

5.2.2 Pairwise Answer Comparison

To assess relative answer quality even when both systems produce plausible outputs, we perform a pairwise LLM-as-judge evaluation.

For each question, we present the judge with:

- the question,
- the ground-truth answer,
- baseline RAG's answer,
- our workflow's answer.

The judge is instructed to choose which answer better matches the ground truth in terms of factual correctness, completeness, and grounding, or to declare a tie if both are equivalent. The instruction template is:

> You are given: (1) a question, (2) a ground truth answer, and (3) two candidate responses (Response A and Response B).
>
> Your task: Decide which response is *better* relative to the ground truth.
>
> Rules:
> - Prefer the response that contains the correct core facts.
> - If both are correct, prefer the one that is more precise, complete, and grounded in the ground truth.
> - If both are equally correct (or equally incorrect), you may declare a TIE.

Over the full 100-question benchmark, we obtain the head-to-head results in Table 3.

Comparison	Ours	Baseline
FinancialQA vs Baseline RAG	**81%**	19%

Table 3: LLM-as-judge pairwise preference between our workflow and baseline RAG on the full benchmark. In 81% of questions, the judge prefers our workflow's answer over baseline RAG's; in the remaining 19%, baseline RAG is preferred.

The pairwise results confirm the correctness numbers in Table 2: in **81%** of questions, our workflow's answer is preferred over baseline RAG, while baseline RAG is preferred in only **19%** of cases. This indicates that even when both answers are nominally acceptable, our workflow tends to produce responses that are more complete, better grounded in the filings, and closer to the annotated ground truth.

5.2.3 Qualitative Error Analysis

Inspection of the judge rationales reveals a consistent pattern:

- When our workflow **wins**, the judge frequently cites:
  - more comprehensive coverage of the relevant section(s),

  - explicit mention of key numeric values or product names,
  - clearer articulation of platform or strategy details (e.g., NVIDIA's platform stack, CUDA, H100 use cases).

- When baseline RAG **wins**, the judge often prefers:
  - slightly more precise phrasing for short factual questions,
  - tighter alignment with a single sentence from the ground truth (e.g., a specific statistic or phrasing),
  - cases where our workflow adds unnecessary context without improving factual content.

Overall, our workflow tends to dominate on:

- concept-heavy questions (e.g., "What does the platform strategy enable across markets?"), and
- multi-sentence explanations (e.g., "How does the computing platform contribute to the markets it serves?"),

Where routing and focused retrieval supply the model with the exact sections that describe platforms, strategies, and architectures.

Baseline RAG's few wins are concentrated in narrow factual lookups (e.g., "What was the turnover rate in fiscal year 2023 for company X?") where any system that retrieves the correct single sentence will succeed and verbosity is slightly penalized by the judge.

Taken together, the accuracy and pairwise results support our main claim: *when document volume is large and filings are structurally similar, baseline RAG becomes fragile, while a routing- and context-aware retrieval architecture substantially improves both correctness and judged answer quality without increasing cost.*

6 Discussion

Our experiments demonstrate that long-document financial QA is fundamentally constrained by two forces: (i) prohibitive LLM context limits and cost profiles, and (ii) retrieval failure modes driven by the high structural similarity across corporate filings. The proposed FinancialQA workflow directly targets both issues. In this section, we interpret the results, outline remaining limitations, and discuss implications for real-world financial analysis systems.

6.1 Why the Workflow Works

The 20-point accuracy gain over baseline RAG is not attributable to any single component, but rather to the *interaction* of several architectural choices. First, the LLM-based routing step dramatically reduces distractors by reasoning over metadata such as company name, year, and filing type. Because 10-Ks share highly repetitive linguistic structure, cosine similarity alone frequently retrieves irrelevant filings. Routing prunes the candidate set before embedding search, producing a much cleaner retrieval pool.

Second, metadata extraction itself plays a nontrivial role. Summaries and document-level attributes allow the router to act as a lightweight reasoning layer over the corpus, approximating what a human analyst would intuitively do when choosing relevant documents.

Third, reranking (to be expanded in future work) provides a mechanism for sharpening chunk selection once routing has restricted the search space. While our preliminary reranking implementation remains simple, we observed that the additional filtering contributed meaningful improvements, especially in sections requiring multi-sentence evidence aggregation. A placeholder remains in our architecture for a learned cross-encoder; completing this is likely to yield even further gains.

Finally, prompting contributed several percentage points of improvement by providing consistent task framing. In enterprise deployments, such increments are meaningful—often the difference between a system that is competitive and one that is unreliable.

6.2 Unexpected Findings

Two empirical results were particularly surprising. First, raw LLM baselines behaved far more naively than expected. Instead of summarizing, pruning, or internally compressing incoming documents, `gpt-4.1` simply consumed the entire PDF token stream, frequently exceeding context limits and incurring extreme cost. This indicates that long-document QA must rely on explicit retrieval mechanisms rather than implicit model behavior.

Second, baseline RAG performed better than expected on short factual lookups. For questions with answers local to a single sentence (e.g., "What was the turnover rate in fiscal 2023?"), baseline RAG sometimes outperformed our workflow because our broader retrieval and slightly more verbose reasoning occasionally diluted the specificity of the response. These cases were rare, but they highlight the importance of controlled verbosity in downstream generation.

6.3 Limitations

The primary limitation of our evaluation is the absence of private or non-public data. Because all filings in our benchmark are public SEC documents, the LLMs may possess partial prior knowledge of their contents, reducing the purity of the evaluation. In practice, however, such prior knowledge vanishes in private-company or due-diligence settings, where context-aware retrieval becomes even more critical.

A second limitation arises in scenarios involving large numbers of short, structurally similar documents. If routing cannot distinguish between filings due to overly uniform metadata, the retrieval pipeline may fall back to similarity-based search over near-identical text fragments, reducing performance. This limitation reflects a broader challenge in multi-document QA: distinguishing semantically similar but distinct sources.

Finally, our current system lacks a learned cross-encoder reranker and uses only similarity search. As discussed above, our architecture includes a placeholder for such a component, and we expect it to meaningfully strengthen the system—particularly for multi-hop questions requiring fine-grained discrimination between passages.

6.4 Future Directions

Several extensions follow naturally from our findings. A learned semantic reranker would likely yield immediate gains, closing remaining gaps in chunk selection. Similarly, a hybrid routing model combining LLM-based metadata reasoning with embedding-based document similarity could increase robustness when metadata alone is insufficient.

Another direction is scaling beyond 15+ documents per query into fully open multi-year, multi-company corpora (50–200 filings). Given the architecture's routing-first design, we expect the workflow to extend naturally to this setting.

Finally, integrating a knowledge-graph layer or table-aware representation may further improve retrieval quality for highly numerical questions, reducing the occasional advantage baseline RAG displayed on narrow factual lookups.

6.5 Broader Implications

The results have direct practical relevance for financial analysis, investment diligence, consulting workflows, and audit review pipelines. Modern financial teams routinely ingest dozens of unstructured documents without the ability to preprocess or reorganize them manually. Raw LLM querying is infeasible due to context limits and economically prohibitive even for moderate workloads. Baseline RAG is inexpensive, but becomes unreliable when structurally similar filings accumulate.

Our workflow provides a path toward accurate, cost-efficient question answering over large and noisy collections of financial documents. The architecture is simple, model-agnostic, and applicable to private-company settings where retrieval-based grounding is not optional but required. Because token usage remains under 20k tokens per query, the system is economically viable for daily industry use.

Overall, our findings demonstrate that *retrieval architecture—not model scale—is the key bottleneck for long-document financial QA*. A routing-aware RAG pipeline yields large performance gains at negligible additional cost, making LLM-based financial analysis dramatically more practical in real-world environments.

7 Conclusion

Long-document question answering over financial filings faces three fundamental barriers: extreme context lengths, high inference cost, and retrieval failure modes triggered by the structural similarity of corporate disclosures. Our results show that existing approaches—raw LLMs or baseline RAG—are either prohibitively expensive, infeasible due to context limits, or highly brittle when many filings are present.

We introduced a multi-stage retrieval architecture based on document-level routing, dense chunk retrieval, optional reranking, and reduced-context LLM querying. This design is simple, model-agnostic, and explicitly optimized for real-world financial analysis where dozens of long documents must be processed simultaneously. Across 100 randomly sampled queries from a 6,900-question corpus, the workflow improves accuracy from 72% (baseline RAG) to 92%, and is preferred in 81% of head-to-head comparisons, while maintaining a token cost comparable to baseline retrieval. These results demonstrate that retrieval architecture—not model size—is the primary limiting factor in financial QA.

Beyond empirical gains, the workflow directly reflects real industry constraints. Financial analysts, diligence teams, auditors, and consultants routinely operate over large, heterogeneous document sets where preprocessing is expensive and private data cannot be supplemented by external web search. Our approach provides a practical and economically viable path for deploying LLM-based analysis in these settings.

Future work includes incorporating a learned cross-encoder reranker, integrating table-aware or knowledge-graph retrieval, and scaling routing to corpora of 100–200 filings. Evaluating the system on private-company documents will further test its ability to operate outside the domain of publicly available data. Overall, our findings highlight that carefully structured retrieval pipelines can make LLM-based financial analysis both accurate and enterprise-ready.

References

Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. In *arXiv preprint arXiv:1908.10063*, 2019.

Kelvin Guu, Kenton Lee, Zora Tung, et al. Realm: Retrieval-augmented language model pre-training. In *International Conference on Machine Learning*, 2020.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *European Chapter of the Association for Computational Linguistics*, 2021.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction. In *Proceedings of the 43rd International ACM SIGIR Conference*, 2020.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, 2020.

Junjie Li et al. Lastingbench: Defending long-context benchmarks against knowledge leakage. In *Findings of the Association for Computational Linguistics*, 2024.

Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. Hoprag: Multi-hop reasoning for logic-aware retrieval augmented generation. 2025.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. In *arXiv preprint arXiv:1901.04085*, 2019.

Yixuan Tang and Yi Yang. Multihop-rag: Benchmarking retrieval augmented generation for multi-hop queries. 2024.

Jingru Wang, Wen Ding, and Xiaotong Zhu. Financial analysis: Intelligent financial data analysis system based on llm-rag. 2025a.

Yifan Wang et al. Fingear: Financial document retrieval and reasoning with domain-specific reranking. In *Findings of the Association for Computational Linguistics*, 2025b.

Antonio Jimeno Yepes, Yao You, Jan Milczek, Sebastian Laverde, and Leah Li. Financial report chunking for effective retrieval augmented generation. Sacramento, CA, USA, 2024. Unstructured Technologies.