**Advanced Multimodal RAG Architecture for ESG Document Intelligence: Integrating Vision Language Models**

**I. Executive Summary: Transitioning to Advanced Multimodal RAG**

The analysis of complex corporate annual reports for environmental, social, and governance (ESG) compliance—specifically carbon accounting and emissions goal tracking—presents a retrieval challenge that transcends the capabilities of conventional, text-based Retrieval-Augmented Generation (RAG) systems. These financial documents, exemplified by detailed analyst presentations and annual filings, are inherently multimodal, embedding critical quantitative evidence—such as Scope 1, 2, and 3 emissions data, financial forecasts, and goal progress—within highly structured tables, informative graphs, and images [cf. user example]. The successful generation of accurate, auditable reports requires a foundational shift in architecture toward Advanced Multimodal RAG.

**A. The Critical Challenge of Multimodal ESG Data**

The primary limitation of basic RAG stems from its reliance on simple text chunking and indexing. This methodology is ill-equipped to handle documents where layout is semantic. When processing complex PDFs that contain data tables, standard text extraction often results in the loss of crucial structural integrity. For example, a parser may concatenate rows and columns into a single, unstructured text blob, thereby destroying the definitive row-column mapping essential for correlating financial metrics or carbon component specifications with their corresponding values.1 This failure mode makes precise quantitative query answering—such as confirming the linkage between a specific product model and its memory capacity, or extracting a precise net-zero timeline from a multi-column target table—impossible to achieve accurately.1

Given that the output of this RAG system is intended for high-stakes financial and ESG assessment, where transparency and accuracy are paramount, the conventional RAG approach is fundamentally insufficient.2

**B. The VLM Imperative for Visual Document Retrieval**

The solution requires adopting architectures centered around **Multimodal Retrieval** and the strategic deployment of Vision Language Models (VLMs).4 VLMs are distinguished by their ability to interpret both visual data (images, layouts) and textual data simultaneously, granting them a holistic understanding of the document page, including charts, tables, and surrounding narratives.5 This capability is instrumental in understanding visual context, such as noting that a specific emission reduction target is presented in a bar chart format rather than a textual list.

The ModernVBERT suite of encoders, highlighted in the user query, is positioned as a specialized tool within this advanced framework. ModernVBERT is engineered as a compact (250M-parameter), high-performance **visual document retrieval encoder** that reportedly matches the performance of models up to ten times larger on visual document benchmarks.7 Due to its architecture, ModernVBERT is optimally suited for integration into the

**encoding and retrieval phases** of the RAG pipeline, maximizing efficiency and speed during the indexing process and enabling sophisticated visual matching capabilities.7 Its strength lies in efficiently encoding the visual and layout semantics required to retrieve the right document page or element.

**II. Foundational Architectural Shift: From Basic to Advanced RAG**

The shift from basic to advanced RAG involves implementing a sophisticated control flow that manages diverse data types and retrieval mechanisms. This complex system is typically structured into three formalized phases: Ingestion, the Inference Pipeline, and Evaluation.4

**A. Deconstructing Advanced RAG: Phases of Implementation**

1. **Ingestion Phase (Offline Processing):** This foundational phase dictates the capabilities of the entire RAG system. It moves beyond simple text chunking to encompass comprehensive data preparation, including multimodal parsing, detailed layout analysis (identifying bounding boxes for tables and charts), generating descriptive captions for visual content, and constructing a robust multi-vector index.4 The primary goal here is to transform the complex PDF corpus into highly structured, semantically rich, and indexable representations.
2. **Inference Pipeline Phase (Online Query Handling):** This phase handles the live user query. It involves a sequence of orchestrated steps: query analysis, routing the query to the optimal data index, executing hybrid retrieval methods, conducting subsequent reranking to increase precision, and finally, using a Large Language Model (LLM) to synthesize the grounded response.4
3. **Evaluation Phase (Quality Assurance):** For mission-critical applications like financial and ESG reporting, continuous evaluation is essential.3 This phase involves systematically testing the system against accuracy benchmarks and developing domain-specific frameworks to ensure high fidelity, usability, and transparency in outputs.3

**B. Hybrid Indexing and Query Routing**

**1. Hybrid Retrieval Necessity**

The heterogeneous nature of ESG reports—which contain long-form narrative text, highly structured financial metrics, and complex visual data—renders reliance on a single, uniform vector index ineffective. Advanced RAG mandates the use of **hybrid indexes** that strategically combine multiple data storage and retrieval methods.4

For instance, when numerical data is extracted from tables and normalized into a structured format (e.g., CSV, JSON), it is often advantageous to store this data in relational databases that support structured querying via SQL.4 This allows for deterministic filtering, aggregation, and precise data retrieval (e.g., "What was the Scope 1 emissions figure for 2023?"). Concurrently, vector stores (such as Chroma or Qdrant) are utilized to store the high-dimensional embeddings generated from textual narratives, image captions, and the multimodal representations created by VLMs like ModernVBERT.9 Combining these retrieval methods ensures that the system can access both fuzzy semantic context and precise structured facts.

**2. Query Routing for Adaptive Retrieval**

A critical feature of the Advanced RAG inference pipeline is the implementation of a **Query Router**.11 Operating as an intelligent agent, the router determines the most appropriate retrieval strategy based on the semantic intent of the user's query. This prevents the system from performing an expensive, imprecise vector search when a precise database lookup is warranted.

For example, if a user asks, "Summarize the introduction section regarding the CEO's commitment," the router directs the query to the text vector index. Conversely, if the query is, "What was the percentage change in operational carbon intensity shown in the table on page 45?" the router can direct the query to the SQL index containing the structured table data. The router uses an initial assessment, often powered by a smaller, fast LLM using function-calling capabilities, to classify the query type and select the optimal retrieval pathway.11

**3. The Semantic Bridge: Enabling VLM-Aware Routing**

Effective query routing in a multimodal context depends on creating a robust "semantic bridge" between the query and the diverse data types stored in the indexes. The text-based Query Router must be explicitly trained or prompted to recognize and manage queries related to purely visual elements (e.g., "Describe the trend illustrated by the largest scatter plot").

To achieve this, the ingestion phase must incorporate rich, VLM-generated metadata for all visual elements, including bounding box coordinates and detailed, descriptive captions.10 The router is then prompted not only with the document's section headings but also with an inventory of available data modalities and their summaries (e.g., "Table 3: Scope 3 Emissions Data," "Figure 5: 5-Year Carbon Intensity Trend Chart"). This comprehensive context allows the router to trigger a specific VLM-based retrieval path when visual keywords are detected, ensuring that visual evidence is accessed directly, rather than relying on an indirect text description that may be inaccurate or incomplete. This orchestration ensures that the visual expertise encoded by ModernVBERT is leveraged by the pipeline's architectural decision-making component.

**III. The Multimodal Ingestion Pipeline: Building the Multi-Vector Store**

The ingestion phase is unequivocally the most complex and critical element of a multimodal RAG system. The quality of subsequent retrieval, and thus the accuracy of the generated reports, hinges on the ability to accurately partition, analyze, and encode the diverse data types present in the annual reports.4

**A. Document Partitioning and Layout Analysis**

Handling complex PDF layouts requires specialized Python tooling to accurately extract and preserve semantic structure, especially for tables and charts.13 Traditional parsers often fail because they treat visual elements as contiguous text, resulting in a loss of relational context.1

Key Python tools and methodologies for addressing this challenge include:

1. **Unstructured.io:** This library is frequently used for initial comprehensive parsing and classification of various document elements (text, tables, images) from complex files.9 It serves as an effective foundation for separating modalities necessary for multi-vector indexing.
2. **PyMuPDF / pymupdf4llm:** For documents like financial reports, preserving table structure is non-negotiable. Libraries optimized for RAG workflows, such as pymupdf4llm, are employed to convert detected table content into semantically rich, structured formats like Markdown.1 This Markdown representation retains the vital row-column relationships that are destroyed by simple text concatenation, ensuring that the input to the embedding model clearly expresses the data structure required for quantitative analysis.
3. **LayoutParser:** Utilizing deep learning models, LayoutParser can accurately identify complex document layouts and determine the precise bounding box coordinates for every extracted element (text chunk, table, image).1 Capturing these coordinates is essential, as they form the backbone of the

**Visual Grounding** mechanism, providing verifiable proof of origin for every retrieved fact.

The overarching goal is to transform the document elements—narrative, structured table data, and visual components—into distinct, highly informative representations, each with rich metadata linking back to its exact location in the source PDF.

**B. Multi-Vector Indexing Strategy and ModernVBERT Integration**

**1. The Multi-Vector Approach**

To overcome the limitations of single-vector indexing, Advanced RAG utilizes a **Multi-Vector Retriever** where a single conceptual element (e.g., a chart) is associated with multiple vectors corresponding to different representations.9 This strategy allows the system to match the user's query style (e.g., a natural language question vs. a structured data request) to the most appropriate indexed representation (e.g., an image caption vector vs. a visual VLM embedding).

**2. The Role of ModernVBERT as Encoder**

ModernVBERT, being explicitly designed for efficient visual document retrieval, is strategically used in this phase.7 The bi-encoder variants (

bimodernvbert or modernvbert-embed) are ideal for generating vectors that simultaneously encode both the visual layout of a page or chart and the associated textual data (e.g., image caption or surrounding paragraphs).7 This creates a holistic, cross-modal embedding that captures the essence of the document element. Integration is straightforward, leveraging the standard

transformers library, with performance optimization achieved via Flash Attention 2 for high GPU throughput.7

**3. The Bimodal Embedding Strategy**

For high-stakes domains like carbon accounting and financial assessment, retrieval accuracy must encompass both visual understanding and deep semantic understanding of industry-specific terminology. ModernVBERT excels visually but is a general VLM encoder.7 Financial-BERT, conversely, has been extensively pre-trained on domain-specific corpora, including corporate reports (10-K, 10-Q) and financial news, granting it superior semantic understanding of financial language.15

Therefore, an optimal **Bimodal Embedding Strategy** must be employed:

* **Visual/Layout Embeddings:** Generated by a **ModernVBERT** bi-encoder for full page images or extracted chart/table crops. These vectors facilitate finding pages based on visual similarity or layout features.
* **Semantic/Text Embeddings:** Generated by a specialized **Financial-Domain Encoder** (e.g., Financial-BERT) for narrative text chunks, descriptive image captions, and structured table Markdown. These vectors ensure accurate matching for financially complex queries.

These disparate vectors are stored cohesively within the vector database, each tagged with metadata defining its source modality (VLM-visual, T-semantic, T-table-markdown). This creates a robust, highly discriminative hybrid index.4

Table 1: Document Partitioning Tools for Multimodal RAG

| **Tool/Library** | **Core Function** | **Primary Output Format for RAG** | **Significance for ESG Reports** |
| --- | --- | --- | --- |
| Unstructured | Full document layout parser, element classification | Text chunks, HTML, JSON, image paths | Efficiently separates modalities for multi-vector indexing and classification. 9 |
| PyMuPDF / pymupdf4llm | PDF parsing, sophisticated table structure preservation | Structured Markdown, Text, Image | Crucial for maintaining table integrity (e.g., carbon calculation data) by avoiding data concatenation. 1 |
| LayoutParser | Deep learning model for layout detection | Bounding Boxes, Element Type | Essential for Visual Grounding and generating precise coordinate metadata for verification of chart/table origins. 1 |

Table 2: Multi-Vector Indexing Strategy for ESG Documents

| **Chunk Type** | **Content Stored** | **Metadata Stored (Crucial)** | **Embedding Model** | **Purpose** |
| --- | --- | --- | --- | --- |
| **Text Narrative** | Standard text chunk (paragraphs, headers) | Page number, section title, bounding boxes (optional) | Financial-BERT/Domain Model 15 | High-precision textual QA and narrative synthesis. |
| **Structured Table** | Markdown representation of the table 1 | Page number, table ID, bounding box | Financial-BERT/Domain Model | Answering precise quantitative questions (e.g., "Total Scope 1 Emissions"). |
| **Visual Caption** | Detailed description of chart/image (VLM generated) 12 | Page number, Image ID, Bounding Box | Financial-BERT/Domain Model | Contextual retrieval using natural language queries about visual concepts (e.g., "What does the graph show about CO2 trends?"). |
| **Full Page/Visual Embed** | Raw page image or chart cropped image | Page number, Document ID | ModernVBERT Bi-Encoder 7 | Visual document retrieval (VDR); finding the most relevant page based on visual layout/content match, independent of textual keyword search. |

**IV. VLM Integration: Positioning ModernVBERT**

ModernVBERT’s value proposition centers on its efficiency and effectiveness in document retrieval tasks.7 While larger, instruction-tuned VLMs can perform the final generation, compact encoders like ModernVBERT are best deployed during the upstream indexing (encoding) and intermediate reranking stages of the pipeline.16

**A. VLM as Primary Multimodal Encoder (Ingestion Phase)**

The primary function of ModernVBERT in the ingestion phase is to generate comprehensive multimodal embeddings. The model (specifically bimodernvbert or modernvbert-embed) is deployed to ingest the document page image along with any associated textual information (if available), producing a single, dense vector representation that inherently captures both visual layout and semantic content.5

This approach mirrors systems like ColPali, which bypasses the often error-prone steps of traditional OCR and layout parsing by embedding the entire PDF page directly.5 The VLM automatically performs modality alignment, drastically improving the retrieval of visually sensitive documents where text content alone is insufficient. This is particularly efficient for large-scale ingestion of hundreds of annual reports, leveraging ModernVBERT’s reported speed and size advantage.7 However, because embedding an entire page reduces granularity, this method is best complemented by the fine-grained text and table indices generated using Financial-BERT, reinforcing the necessity of the Multi-Vector approach.

**B. VLM as a Cross-Modal Reranker (Inference Phase)**

The use of a VLM as a Reranker is a key technique in maximizing the precision of the retrieval phase. After an initial search across the multi-vector index yields a large list of candidate text chunks, table markdowns, and image captions, a Reranker is used to refine this list to the absolute most relevant context.16

ModernVBERT’s late-interaction version, colmodernvbert, is designed for such document retrieval tasks and is suitable for this cross-modal reranking function.7 The mechanism works as follows: the VLM receives the user query, the retrieved textual/semantic metadata, and the original image snippet of the retrieved element. The VLM then cross-compares these modalities, scoring the relevance based on whether the visual evidence (the chart or table structure) actually confirms the textual claim.16 This dramatically increases precision, mitigating a common RAG failure point where a text-based search retrieves text that is merely located

*near* a relevant chart but does not actually describe it. This visual validation step is crucial for high-quality, auditable ESG data extraction.

Table 3: VLM Placement Options in the RAG Pipeline

| **VLM Role** | **Deployment Phase** | **ModernVBERT Variant** | **Benefit for ESG Reports** | **Complexity & Latency** |
| --- | --- | --- | --- | --- |
| **Multimodal Encoder** | Ingestion/Indexing (Offline) | bimodernvbert / modernvbert-embed 7 | Efficient, large-scale indexing of visual and layout context for fast document retrieval. | Moderate (Requires specialized ingestion pipeline). |
| **Multimodal Reranker** | Inference (Online) | colmodernvbert (Late-Interaction) 7 | High-precision refinement; verifies the *visual* relevance of retrieved chunks against the query intent. | High (Adds necessary latency to the retrieval step for quality assurance). 16 |

**V. The Inference Pipeline: Adaptive Retrieval and Context Assembly**

The inference pipeline is the operational core of the Advanced RAG system, responsible for adaptively retrieving evidence and assembling it into a coherent context for the final LLM.

**A. Retrieval and Selection Stages**

1. **Query Analysis and Routing:** As discussed, the Query Router acts first, classifying the user’s need (e.g., numerical data, conceptual summary, or visual trend analysis) and directing the query to the specific index optimized for that data type (e.g., numerical queries directed toward the table Markdown index or SQL database).4
2. **Hybrid Search Execution:** Concurrent searches are executed across the multiple indices. This includes vector searches against both the ModernVBERT visual embeddings and the Financial-BERT semantic embeddings, alongside any necessary SQL lookups for structured data.4
3. **Alignment Optimization:** To enhance the quality of retrieval, an advanced technique involves optimizing the retrieved chunks by matching them to synthetic, context-specific questions generated during the ingestion phase.4 This ensures that the retrieved content is highly relevant and precisely tailored to answer the user's specific information need, improving the retrieval accuracy beyond simple vector similarity.

**B. Context Refinement (Reranking)**

The initial hybrid search typically returns a superset of relevant candidates (50–100 chunks). The Reranker module is mandatory to distill this set into the top 5–10 most informative pieces of context. The VLM Reranker (colmodernvbert) plays a crucial role here, especially for queries concerning visual or tabular data. By visually inspecting the retrieved content against the query, the VLM validates that the retrieved textual elements are contextually grounded in the accompanying chart or table, dramatically reducing false positives and maximizing the signal-to-noise ratio of the final context fed to the generation model.16

**C. Final Context Assembly and Generation**

Once the context has been refined for both semantic and visual relevance, the text chunks, structured table markdown, and image captions (along with crucial bounding box metadata) are collated into the final prompt.

The final Large Language Model (LLM)—which should be a powerful, instruction-following model like GPT-4 or Qwen2-VL, rather than the ModernVBERT retrieval encoder—is then tasked with **Grounded Generation**.16 The LLM is provided with a strict system prompt instructing it to synthesize the report sections or answers

*exclusively* from the provided sources.18 This process ensures the response is factual and verifiable, which is essential for high-stakes ESG analysis.

**VI. Grounded Generation and Output Verification (The ESG Mandate)**

For financial and carbon accounting assessments, the output must be transparent, verifiable, and auditable. It is not sufficient merely to state a fact; the system must demonstrate *where* that fact originated in the source document.2

**A. The Requirement for Visual Answer Grounding**

Advanced RAG moves beyond simple text citations (e.g., "Page 12") to implement **Visual Answer Grounding**. This methodology connects the generated answer directly to its visual origin on the source page.10

The mechanism relies on the bounding box coordinates that were meticulously extracted and stored during the ingestion phase (via LayoutParser/PyMuPDF). When a relevant chunk is retrieved, its metadata includes the page number and the precise coordinates of the information source (the exact paragraph, table cell, or chart region).10 A

**Visual Grounding Module** uses these coordinates to draw a bounding box or highlight the corresponding area on a screenshot of the original source page.10 This resultant visual evidence is presented alongside the generated text. This transparency eliminates the need for manual verification by the human analyst, significantly reducing the verification workload and building trust in the AI-generated output, fulfilling a core requirement for reliable financial assessment.10

**B. Prompt Engineering for Structured Outputs**

The efficacy of the final generation relies heavily on prompt engineering, particularly how the LLM is instructed to handle retrieved structured data. Since tables are retrieved as Markdown, the LLM must be explicitly guided to:

1. **Process Markdown as Data:** Treat the Markdown table not as narrative text to be summarized, but as structured data requiring logical extraction, transformation, and synthesis.
2. **Ensure Explicit Sourcing:** Instruct the LLM to integrate the source metadata (page number, table ID, bounding box coordinates) directly into its output or as part of the citation schema.3

While ModernVBERT ensures the retrieval of the correct visual context, leveraging the semantic expertise of a domain-specific model like Financial-BERT (used in the text embedding phase) ensures that the LLM understands and correctly interprets complex accounting terminology and specific financial nuances present in the data.15

**C. The Agentic Visualization Module: Enhancing Report Generation**

A significant value addition for the user's report generation use case is the integration of an **Agentic Visualization Module**. Reports often require analysts to perform calculations and visualizations based on extracted data (e.g., plotting a 5-year trend of Scope 3 emissions).

Recent studies indicate that LLMs, when provided with appropriate scaffolding (rules and feedback mechanisms), can perform sophisticated data retrieval and autonomously generate necessary visualizations.20 In this Advanced RAG architecture, upon successfully retrieving highly structured data (via the SQL or table Markdown index), the final LLM can be prompted to act as a code generation agent. This agent generates Python code (using libraries like Matplotlib or Plotly) to visualize the extracted data, turning the RAG system from a Q&A tool into a dynamic report generation assistant. This leverages the retrieval accuracy of the system to create high-quality, data-driven figures for the final report.

**VII. Architectural Summary and Implementation Roadmap**

The proposed architecture represents a sophisticated, highly modular system designed to address the challenges of visual and tabular data in ESG documents.

**A. Unified High-Level Architectural Flow**

Table 4: High-Level Architectural Flow for Multimodal ESG RAG

| **Phase** | **Function** | **Core Components & Python Tools** | **VLM/ModernVBERT Role** | **Verifiability Output** |
| --- | --- | --- | --- | --- |
| **1. Ingestion (Offline)** | Multimodal Parsing & Multi-Vector Indexing. | Python (Unstructured, PyMuPDF/pymupdf4llm, LayoutParser); Bimodal Encoder Suite (ModernVBERT + Financial-BERT); Vector Database (Chroma/Qdrant). | **Encoder:** ModernVBERT generates efficient visual/layout embeddings for comprehensive document retrieval. 1 | Rich metadata (Bounding Boxes, Page IDs) stored. 10 |
| **2. Inference: Retrieval** | Query Analysis, Routing, Hybrid Search. | Query Router (small LLM/Function Calling); Vector DB Search (bimodal query); SQL Search. 4 | Minimal (Informs router based on visual tags in index). | Candidate chunks retrieved with source coordinates. |
| **3. Inference: Refinement** | Reranking and Context Optimization. | Cross-Encoder Reranker; VLM Reranker. | **Reranker:** colmodernvbert validates the visual and textual relevance of retrieved chunks against the query. 7 | Highly precise, verified context list. |
| **4. Generation & Output** | Context Synthesis, Grounding, and Report Generation. | Final LLM (e.g., GPT-4); Visual Grounding Module; Agentic Visualization Module. 17 | **Grounding Input:** Supplies visual bounding box metadata to the grounding module for verifiable output presentation. | Generated text linked to highlighted source images. 10 |

**B. Implementation Roadmap for the Data Scientist**

Given the strong background in Python and familiarity with LLMs and RAG, the implementation can be structured into four sequential steps, leveraging open-source frameworks like LangChain or LlamaIndex for orchestration.

**1. Tooling Setup and Data Preparation**

The initial focus must be on establishing the specialized document processing chain. This involves setting up the Python environment, ensuring compatibility with torch, transformers, pillow, and installing layout-aware libraries (unstructured, pymupdf4llm). The desired ModernVBERT variant (e.g., bimodernvbert) and a domain-specific text encoder (e.g., Financial-BERT) should be downloaded and initialized via the HuggingFace transformers library.7

**2. Building the Complex Ingestion Workflow**

A customized parsing loop is required. This loop must iterate through the PDF corpus, using Unstructured and PyMuPDF to extract text, tables (converted explicitly to Markdown format), and image crops. Critically, LayoutParser or a similar tool must be used to capture the precise bounding box coordinates for every extracted element. The Bimodal Embedding Strategy is executed here: ModernVBERT generates visual embeddings for page images, while Financial-BERT generates semantic embeddings for the text and Markdown tables. Finally, the Multi-Vector Store (e.g., Chroma or Qdrant) is populated with these diverse vectors, ensuring the rich metadata, including bounding boxes and modality tags, is preserved.1

**3. Implementing Adaptive Retrieval**

The next step is to introduce the necessary decision-making layers. This involves implementing the Query Router, which uses a smaller, fast LLM to classify query intent and route the search to the appropriate index (e.g., numerical queries to the table index, general queries to the text index, visual queries prioritized via the ModernVBERT index). This is followed by integrating the optional but highly recommended VLM Reranker (colmodernvbert) into the pipeline to perform cross-modal verification on the top-retrieved candidates.11

**4. Establishing Grounded Output and Visualization**

The final step focuses on transparency and utility. The final LLM is configured with a strict system prompt to ensure grounded generation. The **Visual Grounding Module** is developed using Python libraries (e.g., PIL/OpenCV) to programmatically retrieve the source page screenshot and overlay highlights based on the retrieved bounding box coordinates.10 Furthermore, integrating the Agentic Visualization Module allows the system to generate code snippets (and run them) to create custom charts based on the extracted, structured ESG data, directly enhancing the utility for report generation.20

**VIII. Conclusion and Recommendations**

The analysis confirms that the challenge of processing visually rich, layout-sensitive financial reports for ESG analysis necessitates a departure from standard RAG methodologies. The solution lies in implementing an Advanced Multimodal RAG architecture characterized by multi-vector indexing, adaptive query routing, and visual answer grounding.

The compact and specialized **ModernVBERT VLM** is strategically positioned as a highly efficient and effective **retrieval encoder** and **cross-modal reranker** within this architecture. Its primary value is not in text generation, but in its ability to quickly and accurately encode the visual and layout semantics of complex documents, which, when combined with a domain-specific model like Financial-BERT, forms a robust Bimodal Embedding Strategy.

The most critical architectural requirement for high-stakes financial reporting is **verifiability**. By implementing Visual Answer Grounding, which links every generated fact to the precise location (via bounding box coordinates) on the source PDF page, the RAG system achieves the necessary level of transparency and auditability required by ESG analysts and financial domain experts.

The recommended roadmap focuses on building this specialized ingestion pipeline first—the necessary foundation—before layering on the sophisticated retrieval and generation components, ensuring that the resulting RAG system is both high-performance and trustworthy.