Finance RAG: Information Retrieval from Financial Documents

Smit Chandi, Yaksh Shah

June 11, 2025

1 Purpose of Methodology

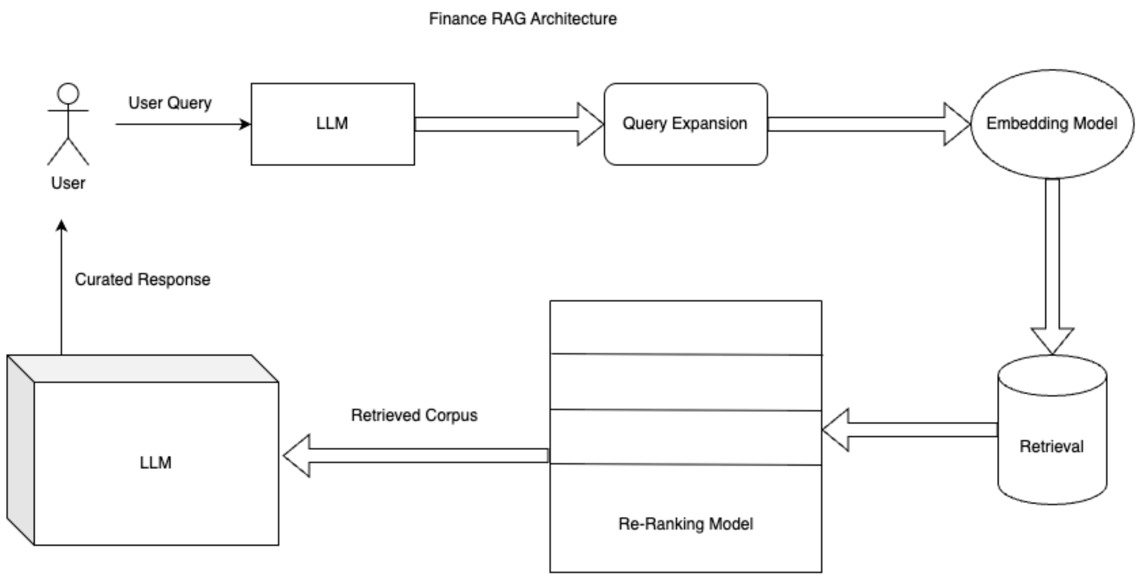

Figure 1: System Workflow

The primary goal of our methodology is to enable accurate and efficient retrieval of relevant information from lengthy and unstructured 10-K financial reports using user-provided natural language queries. These filings are often dense, verbose, and domain-specific, making it imperative to adopt a robust semantic search and response framework.

Effective Problem Solving through Chosen Techniques

To address this, we designed a multi-stage pipeline consisting of the following components:

- **Recursive and Agentic Chunking**: This enables segmentation of lengthy documents into semantically meaningful units while maintaining contextual continuity, which is essential for high retrieval accuracy.
- **Transformer-Based Embedding Models**: By fine-tuning models like `bge-m3` and `gte-multilingual-base`, we ensure that chunk embeddings are highly aligned with the semantics of financial domain queries.
- **Vector Store with Metadata**: Embeddings are stored in Pinecone along with rich metadata (e.g., company, sector, year), which allows for advanced filtering and dynamic context retrieval.
- **LLM Integration for Structured Responses**: Retrieved chunks are passed to an LLM (e.g., LLaMA-17B, Gemini) that can interpret them and return human-like, structured responses to complex questions.

Each of these steps contributes directly to improving the precision and usability of the overall system by ensuring that document understanding and response generation are tightly coupled.

Necessity of Model Comparison

We conducted empirical comparisons between multiple embedding models to validate performance gains through domain-specific fine-tuning. The evaluation metrics (e.g., NDCG@10, Precision@1, MRR@10) helped us quantitatively assess how well each model retrieves relevant document chunks.

The model comparison was crucial for the following reasons:

- To determine which architecture generalizes best across financial language.
- To quantify the impact of LoRA-based fine-tuning on retrieval effectiveness.
- To inform final deployment decisions in terms of trade-offs between latency and accuracy.

These insights not only guided the model selection but also validated the end-to-end methodology for scalable and high-fidelity information retrieval from 10-K filings.

2 Problem Statement

Today, we have a huge amount of unstructured data, especially in the financial world. This includes things like company reports, news articles, and other documents that aren't neatly organized. We need a way to use this data and understand what it means. Large Language Models, or LLMs, are very helpful here because they can understand and work with text.

But LLMs have a problem: they can sometimes make things up, which we call hallucination. This is a big issue, especially in finance where being accurate is really important. To stop this, we use something called RAG (Retrieval-Augmented Generation). RAG systems use the unstructured data we have and give it to the LLM as background information, or context. The LLM then uses this context to create an answer based on the question asked, making the answer more reliable and less likely to be made up.

However, there's a main challenge with RAG. LLMs can only handle a limited amount of text at once, which is called their context length. Financial reports are often very long and contain many tables with lots of numbers and other figures. These documents are too big to fit into the LLM's context all at once.

Our system is designed to solve this problem. It will take these big financial documents and break them down into smaller pieces, or chunks. It will then be able to understand complex questions about finance. Based on the question, it will find and pull out (retrieve) the most relevant pieces of information from the document chunks. Finally, it will use this found information to give the user a clear answer. Our main goal for this system is to get better at finding the right pieces of information (improving retrieval accuracy) based on what the user asks. We will study how different models and methods help us do this by doing tests to see which ones work best.

3 Data Collection and Preparation

The dataset will primarily consist of 10-K annual reports from publicly listed companies. We plan to target 20–30 companies across major industry sectors such as Technology, Finance, Healthcare, and Consumer Goods to enable sector-wise comparative analysis.

1. Planned Data Extraction Pipeline

We intend to build an automated ETL pipeline to retrieve 10-K filings from the SEC EDGAR database using API.

2. Planned Chunking Strategy (Recursive + Agentic Chunking)

To process the lengthy and hierarchical structure of 10-K filings, we aim to implement a hybrid chunking strategy:

- **Recursive Text Splitting:** Initially, the documents will be split using a recursive character- or token-based approach to respect natural section boundaries.
- **Agentic Chunking (Planned):** We also plan to integrate an agentic layer that uses a lightweight LLM to semantically evaluate and optimize chunk boundaries. This is particularly useful for capturing coherent blocks from sections like "Financial Performance" or "Management's Discussion and Analysis."

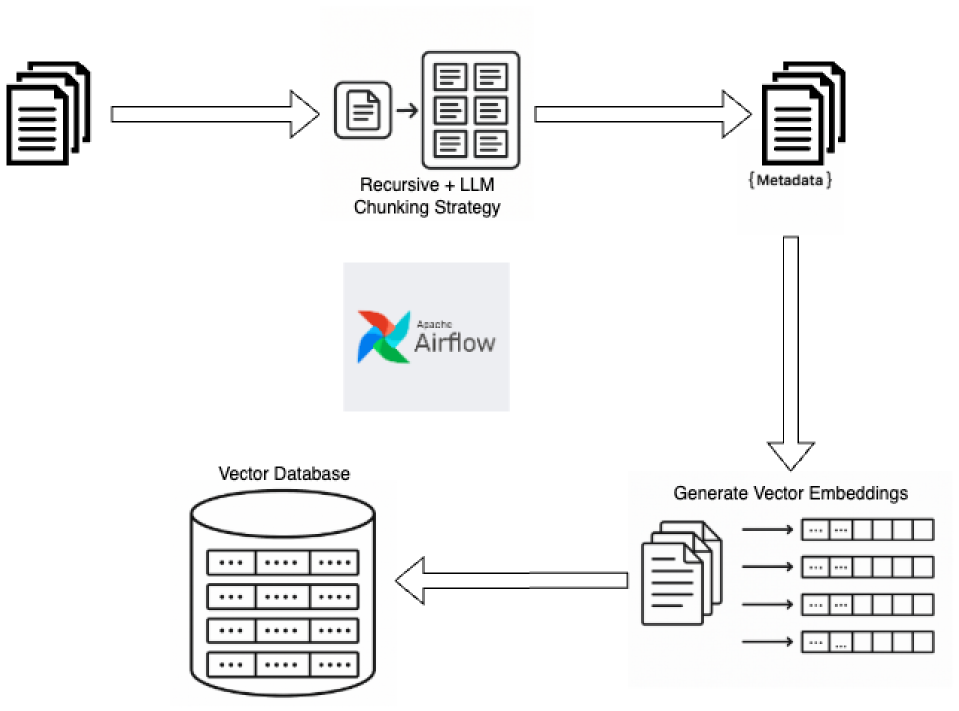

Figure 2: Data Pipeline

3. Embedding Generation

We will be using the our own finetuned model to encode each chunk into a high-dimensional vector. These embeddings will represent the semantic content of each document section, enabling robust similarity-based search and downstream applications such as question-answering.

4. Vector Storage in Pinecone

All generated embeddings will be stored in a **Pinecone** vector database. Each vector will be indexed along with its associated metadata, allowing:

- **Efficient semantic retrieval** using vector similarity.
- **Metadata-based filtering** (e.g., by company, sector, or year).
- **Scalability** across large volumes of 10-K reports.

4 Selection of LLM's

4.1 Embedding Models

To generate dense vector representations of the chunked 10-K filings, we explored transformer-based embedding models capable of capturing semantic relationships in long-form financial text.

Model Consideration

We selected two state-of-the-art embedding models for experimentation:

- **BAAI** `bge-m3`: A powerful English-language embedding model optimized for multi-granularity retrieval tasks.
- **Alibaba-NLP** `gte-multilingual-base`: A multilingual embedding model that generalizes well across domains and supports cross-lingual retrieval.

Both models were fine-tuned on our domain-specific dataset using contrastive learning objectives to better align chunk embeddings with retrieval intents.

Evaluation Metrics

We evaluated each model using a range of ranking and retrieval-based metrics:

- **Cosine Accuracy@10**: Measures the accuracy of retrieving at least one relevant result in the top-10 candidates.
- **Precision@1**: Fraction of relevant results in the top 1 returned.
- **Recall@10**: Proportion of relevant results retrieved among the top 10.
- **NDCG@10 (Normalized Discounted Cumulative Gain)**: Captures the ranking quality by penalizing lower-ranked relevant documents.
- **MRR@10 (Mean Reciprocal Rank)**: Measures the average of reciprocal ranks of the first relevant document.
- **MAP@100 (Mean Average Precision)**: Average precision across the top 100 retrieved items.

4.2 Large Language Models (LLMs)

To generate structured, context-aware responses from the retrieved document chunks, we plan to integrate multiple large language models into the system. These LLMs will be responsible for interpreting the semantic vectors retrieved from the Pinecone database and formulating coherent answers to user queries. Rather than relying on a single model, our design allows flexibility for end-users to select from a pool of high-performing LLMs based on their preferences for response quality, latency, or model architecture.

Model Candidates

The following models are under consideration for integration:

- **LLaMA-17B**: A high-parameter open-weight transformer model from Meta, optimized for long-context reasoning, summarization, and retrieval-based QA tasks.
- **Gemini**: A proprietary model by Google known for its advanced reasoning capabilities and multi-modal alignment, suitable for structured document parsing.
- **Additional Models**: We also plan to explore one or two additional models such as `Mistral` based on cost-performance trade-offs and API integration feasibility.

System Flexibility

The architecture will allow users to choose their preferred LLM at query time. This dynamic selection enables:

- **Comparative analysis** of responses from different models.
- **Customization** based on latency, cost, or factual accuracy.
- **Improved robustness** by offering fallback models in case of API failures or degraded performance.

This modular and agent-friendly approach ensures that the system remains adaptable to future model upgrades and experimentation with new LLMs as they become available.

5 Model Development and Training

5.1 Architecture and Configuration

We focused on fine-tuning transformer-based embedding models for semantic chunk representation of 10-K filings. Traditional feature-based ML models were not applicable due to the unstructured nature and length of financial text.

Transformer Embedding Models

The selected models were:

- **BAAI bge-m3**: An English-focused transformer model pre-trained for multi-granularity retrieval tasks.
- **Alibaba-NLP gte-multilingual-base**: A multilingual transformer embedding model pre-trained on a wide corpus for cross-lingual representation.

Both models follow the standard transformer encoder architecture, consisting of:

- **Multi-head self-attention layers** to capture contextual relationships between tokens.

- **Feed-forward neural networks** with residual connections.
- **Layer normalization and positional encodings**.

LoRA (Low-Rank Adaptation) was used to fine-tune the models efficiently. LoRA injects trainable low-rank matrices into the attention layers, significantly reducing the number of parameters that need updating, while preserving performance.

5.2 Training Process

The dataset used for training consisted of chunked segments of 10-K reports from selected companies. We used a contrastive learning objective where each anchor chunk was paired with a relevant positive and irrelevant negative sample.

- The data was split into:
  - 80% training set,
  - 10% validation set,
  - 10% test set.
- For, The training process used Multiple Negatives Ranking Loss.
- Early stopping was applied based on validation loss to prevent overfitting.

5.3 Hyperparameter Tuning

Key hyperparameters such as learning rate, LoRA rank, warmup steps, and batch size were tuned using a grid search strategy on the validation set.

- **Learning rate**: `[2e-5, 5e-5, 1e-4]`
- **LoRA rank**: `[4, 8, 16]`
- **Batch size**: `[16, 32]`
- **Number of epochs**: Up to `10`, with early stopping.

Evaluation during tuning was based on NDCG@10, Precision@1, and Recall@10 scores. The best-performing configuration was selected for final model export and embedding generation.

Transfer Learning Strategy

Rather than training from scratch, we adopted a transfer learning strategy by leveraging pre-trained encoder models and applying LoRA-based fine-tuning. This significantly reduced computational cost while still achieving meaningful domain adaptation.

Results and Final Selection

The following table summarizes the performance gains observed after fine-tuning:

bge-m3 Results (Top model):

Metric	**Base Model**	**Fine-tuned Model**	**Gain**
cosine_accuracy@10	0.8276	0.8652	+3.76%
cosine_precision@1	0.6088	0.6630	+5.41%
cosine_recall@10	0.8276	0.8652	+3.76%
cosine_ndcg@10	0.7117	0.7618	+5.01%
cosine_mrr@10	0.6753	0.7290	+5.37%
cosine_map@100	0.6801	0.7336	+5.35%

gte-multilingual-base Results:

Metric	**Base Model**	**Fine-tuned Model**	**Gain**
cosine_accuracy@10	0.7901	0.8508	+6.08%
cosine_precision@1	0.5536	0.6398	+8.62%
cosine_recall@10	0.7901	0.8508	+6.08%
cosine_ndcg@10	0.6707	0.7449	+7.42%
cosine_mrr@10	0.6325	0.7109	+7.84%
cosine_map@100	0.6387	0.7156	+7.70%

While both models showed significant improvement post fine-tuning, the `bge-m3` model was selected as the final embedding model due to its consistent performance and alignment with the domain-specific context of financial documents.