Multi-Agent QA on Financial Documents

Prepathon 2025 NLP Project

Saksham Madan
IIT BHU (Varanasi)
Mathematics and Computing, Part II

October 7, 2025

Abstract

This project implements a multi-agent question answering (QA) system over financial documents. The system consists of two tasks: (i) retrieval of financial report content using Jina embeddings with FAISS indexing, and (ii) dynamic orchestration of multiple specialized agents under the supervision of a Groq LLM. This report documents the architecture, implementation, usage, and deliverables.

1 Introduction

Financial documents contain structured and unstructured information, including text, tables, and numerical data. Extracting insights requires both retrieval and reasoning. This project explores a multi-agent pipeline where specialized agents collaborate to answer user queries.

2 System Architecture

The system is composed of two layers:

1. **Retriever Layer**: Converts PDFs to page images, embeds them using Jina AI, builds FAISS index, retrieves top-k relevant pages, and clusters them for contextual summaries.

2. **Agent Layer**: A swarm of specialized agents (Retriever, Table, Math, Summarizer) coordinated by Groq Supervisor.

2.1 Architecture Diagram

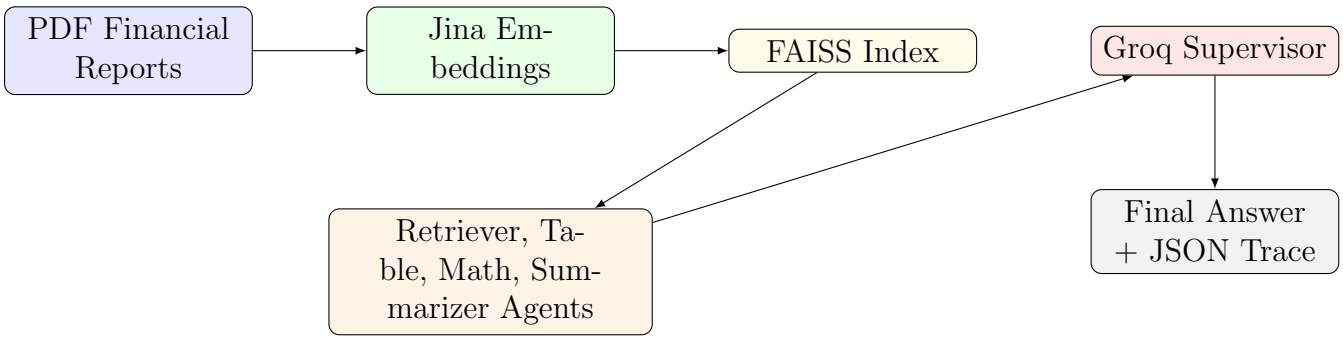

Figure 1: System Architecture: Retrieval + Multi-Agent QA

3 Project Tasks

3.1 Task 1: Document Retriever

- Convert PDF → Images
- Embed images using **Jina Embeddings**
- Build a **FAISS index** for retrieval
- Retrieve relevant pages and parse them
- Apply RAPTOR-style clustering

3.2 Task 2: Multi-Agent QA

- Agents: Retriever, Table, Math, Summarizer
- **Groq LLM** acts as supervisor
- Dynamically selects next agent
- Produces final answer with reasoning trace

4 Project Structure

```
.
retriever.ipynb     # Task 1: Retrieval pipeline
multi_agent.ipynb   # Task 2: Multi-agent orchestration
data/               # Embeddings, indexes, memory
outputs/            # JSON traces
requirements.txt    # Dependencies
.gitignore
report.pdf
README.md
```

5 Installation

5.1 Clone Repo & Install Dependencies

```
git clone https://github.com/s4kr3d-w0r1d/FinanceRAG.git
cd FinanceRAG
pip install -r requirements.txt
```

5.2 Environment Variables

```
export JINA_API_KEY="your_jina_key_here"
export GROQ_API_KEY="your_groq_key_here"
```

6 Usage

6.1 Task 1: Retriever

1. **PDF to Images**: Converted at 300 DPI using `pdf2image`.
2. **Jina Embeddings**: Page-level image embeddings stored in FAISS.
3. **FAISS Search**: Cosine similarity search for top-k pages.
4. **Page Parsing**: Using `unstructured` library and OCR fallback (Tesseract).
5. **RAPTOR-like Clustering**: Text chunks clustered with Agglomerative Clustering + summarization.

6.2 Task 2: Multi-Agent QA

- **Retriever Agent**: Searches FAISS and returns context.
- **Table Agent**: Heuristically extracts financial tables (using regex + pandas).
- **Math Agent**: Computes YoY growth and ratios from tables.
- **Summarizer Agent**: Uses Groq LLM to generate a natural language answer.
- **Groq Supervisor**: Dynamically orchestrates agent execution until STOP.

7 Example Query

```
query = "Compare the YoY revenue growth and R&D spending
         between 2021 and 2022, and summarize the risks
         affecting future revenue."
result = system.run(query)
print(result["final_answer"])
```

8 Deliverables

- Working code in both notebooks
- JSON logs of agent traces in `/outputs/`
- Report including architecture, sample runs, and evaluation

9 Notes

- Table extraction is heuristic and depends on PDF structure.
- Groq is used as supervisor and summarizer.
- Jina embeddings handle page-level retrieval.

10 Conclusion

The project demonstrates a practical application of retrieval-augmented generation and multi-agent orchestration for financial document analysis. By combining embedding-based retrieval and reasoning with dynamic agent control, the system provides interpretable, structured answers to complex queries.