WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

FINRAG: A LOCAL RETRIEVAL-AUGMENTED GENERATION SYSTEM FOR FINANCIAL DOCUMENT QUESTION ANSWERING

S Dhinakaran[1], Sujith S[2], Vishal L[3], Thavasi Perumal D[4], Vishwa K[5]

Artificial Intelligence and Data Science (AI&DS)
Sri Shakthi Institute of Engineering and Technology
Coimbatore, Tamilnadu, India.
E mail: sujithsugumaran23@gmail.com

ABSTRACT :

Financial reports contain a large amount of useful information, but finding one particular value in a long filing can be time-consuming. This project presents FinRAG, a simple Retrieval-Augmented Generation system designed for financial documents. A user can upload a financial PDF, such as a company 10-Q report, and ask questions about its contents in plain English. The system first extracts the text from the PDF and divides it into smaller chunks. The chunks are converted into embeddings and stored in a vector database. When the user asks a question, the system finds the most relevant parts of the document and gives them to a locally running Llama 3 model through Ollama. The final response is then shown in a web interface and can include Markdown tables. The main aim of the project is to make financial document analysis easier while keeping the system local and based on free, open-source tools.

Keywords: *Retrieval-Augmented Generation, Financial RAG, Llama 3, Ollama, Vector Database, Semantic Search, Financial Documents, Question Answering.*

I. Introduction

Financial filings are important sources of information for understanding a company's performance and business activities. They contain revenue figures, expenses, assets, liabilities, risks, management discussions, and many other details. The difficulty is not usually finding the report; it is finding the exact information needed inside a report that may contain hundreds of pages.

A person can use the PDF search option to look for a word, but keyword search is not always enough. For example, a user may ask, "What are the total revenues?" while the report may describe the same information using slightly different wording. A system that understands the meaning of the question can make this process easier.

FinRAG addresses this problem by combining document retrieval with a language model. Instead of asking the model to answer from memory, the system first searches the uploaded document and then uses the retrieved text as context. This helps keep the response connected to the document selected by the user.

II. Problem Statement

Reading a long financial filing to find a small piece of information can take a lot of time. Users may also need to compare information from different sections of the same report. Normal keyword search can miss useful passages when the wording of the question and the wording in the document are different. Therefore, the project aims to build a system that can understand a user's question, retrieve the relevant information from a financial PDF, and present the answer in a simple natural-language form.

III. Objectives

The main objective of this project is to develop a local RAG system for asking questions about financial PDF documents. The project focuses on the following objectives: accepting and validating PDF files; extracting their text; splitting the text into useful chunks; creating embeddings; storing the chunks in a vector database; retrieving relevant content for a question; generating an answer with Llama 3 through Ollama; and displaying the answer clearly in a web interface.

IV. Literature Survey

Retrieval-Augmented Generation has become a useful approach for question answering over external documents. The basic idea is straightforward: first

find useful information from a knowledge source, and then give that information to a language model before generating the answer. This is especially useful when the required information is specific to a document and may not be present in the model's original training data.

Vector-based retrieval is useful because it looks at the meaning of text rather than depending only on exact word matches. A financial question and the relevant paragraph may use different phrasing, but their meanings can still be similar. Embeddings allow the system to represent both the question and document chunks in a form that can be compared.

In this project, Llama 3 is used as the language model and is run locally through Ollama. This keeps the generation part of the application on the local machine and avoids making the project dependent on a paid external LLM service. The complete workflow therefore connects PDF processing, semantic retrieval, vector storage, and local text generation.

V. Proposed System

The proposed system has two main parts: a web frontend and a backend service. The frontend is built using plain HTML, CSS, and JavaScript, so no frontend framework is required. It provides the file upload area, question box, answer area, loading messages, and error messages.

The backend provides two main API endpoints. The POST /upload endpoint receives the PDF, processes it, and stores the resulting chunks. It returns the filename, number of chunks stored, and a collection name. The frontend keeps this collection name and sends it with later questions.

The POST /ask endpoint receives the user's question and collection name. It retrieves relevant chunks from the selected document and sends the retrieved context to Llama 3 through Ollama. The generated answer is returned to the browser. Since local inference can take some time, the interface shows a loading state and disables the button while the request is running.

VI. System Architecture

The system follows a step-by-step pipeline. First, the uploaded PDF is checked and its text is extracted. The text is then split into smaller chunks. Each chunk is converted into an embedding and stored in a vector database. When a question is asked, the system searches the corresponding collection and selects the most relevant chunks. These chunks are given to Llama 3 as context, and the model produces the final answer.

Stage	Input	Process	Output
Upload	Financial PDF	Validation	Accepted document
Extraction	PDF	Text extraction	Document text
Chunking	Text	Splitting	Retrievable chunks
Embedding	Chunks	Embedding model	Vectors
Storage	Vectors + text	Vector DB	Searchable collection
Retrieval	Question	Semantic search	Relevant context
Generation	Question + context	Llama 3 / Ollama	Answer

TABLE I. MAIN STAGES OF THE FINRAG SYSTEM

VII. Methodology

A. Document Upload and Validation

The process starts when the user selects a PDF file from the browser. The frontend sends the file to POST /upload using multipart/form-data. The backend accepts only PDF files. If the selected file is not a PDF, the user receives a clear error message. After successful processing, the backend returns the number of chunks stored and the collection name.

B. Text Extraction and Chunking

The text is extracted from the uploaded document and divided into smaller pieces. Keeping the whole report as one block would make retrieval less useful and would create unnecessarily large model input. Chunking allows the system to search smaller sections and bring back only the parts that are useful for the current question.

C. Embeddings and Vector Storage

Each chunk is converted into a numerical representation called an embedding. The same type of representation can be created for the user's question. The vector database then makes it possible to find chunks whose meaning is close to the question. The chunks are stored under a collection name associated with the uploaded document.

D. Retrieval and Context Construction

When the user submits a question, the backend searches the selected collection. The most relevant chunks are collected and used as context for the language model. This is the main RAG step because the model receives information taken from the uploaded document before it generates the answer.

E. Local Answer Generation

The retrieved context and the user's question are passed to Llama 3 through Ollama. The model generates an answer in natural language. The backend returns the answer as JSON, and the frontend renders Markdown so that tables and other structured content are easy to read.

VIII. Workflow

The complete workflow starts with a financial PDF and finishes with an answer to the user's question. Once the PDF has been processed, the returned collection name is reused for later questions during the active server session.

1. Upload Financial PDF
2. Extract Text
3. Split into Chunks
4. Create Embeddings
5. Store in Vector Database
6. Retrieve Relevant Chunks
7. Llama 3 through Ollama
8. Generate Answer

FIGURE 1. END-TO-END FINRAG WORKFLOW

IX. Web Portal and User Interaction

The web page is kept intentionally simple. The upload section contains a PDF file picker, an Upload button, and a status message. After the document is processed, the page shows how many chunks were stored and keeps the collection name automatically.

The question section contains a text box and a Get Answer button. The button is enabled only after a document has been uploaded successfully. During upload or answer generation, the relevant button is disabled and a loading message is shown. This prevents accidental repeated requests while the backend is still working.

The answer area uses a free Markdown library such as marked.js. This means a response containing a Markdown table can be displayed as a proper table instead of showing the Markdown symbols. The page can also keep a small list of questions asked during the current session so that the user can easily review earlier questions.

X. API Design

A. Upload API

POST /upload accepts a multipart/form-data request. The form field must be named file. After successful processing, the response contains filename, chunks_stored, and collection_name. The returned collection name is required when asking questions.

B. Question Answering API

POST /ask accepts JSON containing text and collection_name. The text field contains the user's question, while collection_name identifies the uploaded document. The response contains an answer string, which may include Markdown.

C. Interactive Documentation

The backend provides interactive API documentation at http://127.0.0.1:8000/docs. The API base URL is http://127.0.0.1:8000. These pages are useful for testing the backend endpoints before connecting them to the frontend.

XI. Error Handling and Reliability

The frontend should give the user a useful message for each common failure. A 400 error is shown when a non-PDF file is uploaded. If the server returns a 500 error because the document collection is no longer available after a restart, the page should tell the user to upload the document again. If the backend cannot be reached at all, the page should explain that the FastAPI server needs to be running.

These messages are important because the application performs processing locally and the user may otherwise see only a generic browser error. Clear feedback makes the system easier to demonstrate and easier to troubleshoot.

XII. Computational Resources

The project is designed around free and open-source components that can run locally. The amount of time needed depends on the PDF size, number of chunks, embedding process, vector search, and Llama 3 inference. A reasonably modern processor and enough RAM are recommended. On a CPU-only machine, answer generation can be slower, especially for larger models.

The system also needs storage for the uploaded PDFs, vector data, model files, and application files. A regular laptop can be used for development and demonstration when the selected model and documents fit within the available hardware.

XIII. Results and Discussion

The completed design provides a complete path from document upload to question answering. A user can upload a financial filing and then ask several questions about that document without processing the PDF again for every question. The retrieval stage supplies document-specific context to the language model, while Llama 3 converts that context into a readable answer.

One useful feature of the system is the separation between document processing and question answering. The PDF is processed once during upload, and the resulting collection is reused for later questions in the same server session. This makes the interaction more natural for a user who wants to explore different parts of the same financial report.

The local Llama 3 and Ollama setup also fits the project's goal of avoiding a paid external LLM service. The approach is suitable for an academic prototype because the main RAG stages are visible and can be tested separately: extraction, chunking, embeddings, retrieval, and generation.

XIV. Advantages

The system is easy to use because the user only needs to upload a document and type a question. It can retrieve information based on meaning rather than relying only on exact keywords. The use of a local language model reduces dependence on an external paid API. The modular design also makes it possible to replace the embedding model or vector database later without changing the basic user workflow.

XV. Limitations

The quality of the final answer depends partly on the quality of the text extracted from the PDF. Scanned pages, unusual layouts, images, and complicated financial tables may not be handled well without additional OCR or specialized table extraction. Retrieval quality also depends on the chunking and embedding choices.

Another limitation is local inference speed. Since Llama 3 runs through Ollama on the user's machine, CPU-only systems may take longer to generate answers. In addition, the current upload index is stored in memory. If the backend is restarted, the document association is lost and the user has to upload the PDF again.

The generated response should also be treated as an aid for reading the document rather than as a replacement for professional financial, accounting, legal, or investment review. Important numbers should be checked against the original filing.

XVI. Future Enhancement

The project can be improved in several practical ways. Future versions can show the page or section from which an answer was retrieved, highlight the supporting passages, and improve extraction of financial tables. OCR can be added for scanned documents. Persistent storage can allow users to return to previously indexed reports without uploading them again.

The system can also be extended to support multiple documents, conversation history, authentication, access control, and evaluation datasets. An evidence indicator showing the retrieved passages would make the answers easier to verify and give users more confidence when working with financial figures.

XVII. Conclusion

FinRAG offers a practical way to ask questions about financial PDFs using a combination of document processing, semantic retrieval, vector storage, and local language-model generation. The user does not have to manually search through every section of a long filing. After uploading the document, the system prepares it for retrieval and uses the relevant sections to answer later questions.

The project shows how Retrieval-Augmented Generation can be used for a focused financial-document application. Using Llama 3 through Ollama keeps the generation process local, while the simple web frontend makes the system easy to demonstrate. With improvements such as source citations, better table extraction, persistent indexing, and systematic evaluation, the prototype can be developed into a more reliable financial document assistant.

XVIII. REFERENCES

[1] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems, vol. 33, 2020.

[2] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," Proceedings of EMNLP-IJCNLP, 2019.

[3] T. Brown et al., "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems, vol. 33, 2020.

[4] Meta AI, "Llama 3: Meta's Open Foundation and Fine-Tuned Chat Models," 2024.

[5] Ollama, "Get up and running with large language models locally," project documentation.

[6] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," IEEE Transactions on Big Data, 2019.

[7] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," Foundations and Trends in Information Retrieval, vol. 3, no. 4, 2009.