An Investigation on RAG Question-Answering System Implementation

Jiayi Yan*

College of Liberal Arts & Sciences, University of Illinois Urbana-Champaign, Champaign, the United States

Abstract. This paper presents the design and implementation of a Retrieval-Augmented Generation (RAG) question-answering system based on a Streamlit web application. The system allows users to upload one or more PDF files and ask questions in natural language. It retrieves and indexes PDF content using FAISS for semantic vector search and BM25 for keyword-based retrieval, combining both methods to improve accuracy. A cross-encoder is used to rerank the retrieved results, ensuring that the most relevant passages are used to generate the final answer. The OpenAI language model then produces short and clear responses grounded in the retrieved context. The system also supports notes generation, multilingual interfaces in English and Chinese, and memory for continuous dialogue. Experiments conducted on academic PDFs show that hybrid retrieval with reranking performs better than single retrieval methods, giving more precise and meaningful answers. The Streamlit interface and Cloudflare tunnel make the system easy to deploy and share. This project demonstrates how RAG technology can effectively extract information from complex documents and provide practical tools for education, research, and enterprise applications.

1 Introduction

With the exponential growth of digital documents, retrieving relevant information from large text corpora has become increasingly important. Traditional search methods such as keyword matching often fail to capture the semantic meaning of text, especially in complex documents like PDFs. Recent advancements in machine learning, particularly in the areas of dense retrieval and cross-encoders, have significantly improved information retrieval systems [1]. These methods combine traditional term-based search algorithms like BM25 with modern dense retrieval techniques, enabling a more accurate understanding of context and improving retrieval performance [2].

FAISS [3], a popular library developed by Facebook AI, facilitates efficient similarity search over large datasets by providing high-performance indexing for dense vector representations. By indexing PDF content semantically, FAISS helps retrieve relevant segments even when exact keywords are not present. The cross-encoder, another critical

* jiayi22@illinois.edu

component, re-ranks the retrieved segments to ensure that the most relevant context is presented to the user. Finally, the OpenAI chat model is employed to generate answers that are both contextually relevant and concise, ensuring a high-quality user experience. In this project, the paper integrates these techniques into a Streamlit app, enabling users to interact with the system through a simple web interface. The goal is to demonstrate how combining traditional and modern retrieval techniques can provide an efficient solution for answering questions from complex documents.

2 Related Works

Retrieval-Augmented Generation (RAG) technology is emerging as a core solution to tackle the pain points in traditional data services. The PDF question-answering system mentioned earlier has verified its practical value through technical implementation, which deeply aligns with the actual needs of enterprise business data supply and archive retrieval. In enterprise business scenarios, traditional data supply relies heavily on technical personnel to write SQL scripts. This not only results in low response efficiency but also limits business personnel's ability to conduct independent data analysis due to "technical barriers" [4].

In the field of archive retrieval, traditional keyword-matching models face issues such as "incomplete knowledge" (missing relevant resources), "unfamiliarity for new users" (complex operations), and "cross-language barriers," which severely restrict the development and utilization of archive resources [5]. The practical value of RAG technology perfectly matches the needs of these scenarios. In enterprise settings, through natural language interaction functions, business personnel can directly obtain data by asking questions, significantly reducing reliance on technical personnel and improving response efficiency [4]. In the archive field, semantic retrieval capabilities effectively solve the "incomplete knowledge" problem; natural language interaction lowers the operational threshold for new users; and hybrid retrieval strategies support cross-format resource integration, comprehensively optimizing the retrieval experience [4].

3 Method

3.1 PDF Parsing Module (parse pdf)

The PDF parsing module is responsible for converting uploaded PDF files into a standardized data structure for subsequent processing [6-8]. It uses the pypdf library to read file content and extract document metadata such as title and author, substituting the file name or an empty string if these are missing. The parsing process iterates through each page, extracting text and skipping blank pages to ensure that no invalid data is introduced into later retrieval steps. The output is a list of Document objects containing both text content and metadata, providing a unified input format for later splitting, vectorization, and retrieval.

3.2 Paper Note Generation Function (run summary for doc)

The paper notes generation function targets the full content of a single paper to produce a structured long-form note [9, 10]. It concatenates all page texts into a context (limited to 15,000 characters to avoid exceeding the model's context window) and combines this with document metadata in a predefined prompt template. Using LangChain's chain mechanism (PromptTemplate → LLM → OutputParser), it outputs a "Research Core" note divided into Content, Innovations, and Shortcomings. This module ensures consistent note structure and supports both Chinese and English modes.

3.3 Question Answering Function (runqa)

The question-answering function is designed to retrieve the most relevant document fragments from the vector retriever based on a user's query and generate an answer strictly grounded in the provided context. It first calls retriever.get_relevant_documents to obtain the top-K relevant chunks, then incorporates short-term conversation memory (the last three QA pairs) to maintain contextual continuity. The concatenated context is limited to 7,000 characters to prevent truncation by the model. This module is suitable for instant interactions such as querying paper details or explaining concepts.

3.4 Vector Retrieval Construction Module

The vector retrieval construction module is responsible for splitting parsed PDF text into suitable segments and building a searchable vector index. It prioritizes using SemanticChunker for semantically coherent boundaries, and falls back to RecursiveCharacterTextSplitter (chunk size=1200, overlap=200) if semantic splitting fails. The text is then converted into vectors using OpenAIEmbeddings and stored in a FAISS vector database. The retriever interface is exposed via .as_retriever, multiple.

3.5 Multilingual UI and Prompt Management (LANGMAP)

The multilingual UI and prompt management module centralizes the management of Chinese and English interface text and prompt templates, allowing users to switch languages with one click. All buttons, messages, warnings, and prompt templates are stored in LANGMAP, and the system dynamically loads the corresponding UI text and prompt content after a language switch. This design ensures consistency between the interface and generated content and facilitates future expansion to more languages or prompt style adjustments.

3.6 Session State Management (st.sessionstate)

Session state management stores key objects and data across interactions to avoid repeated initialization and maintain conversation continuity. Stored items include the large language model instance (llm), embedding model instance (emb), retriever instance (retriever), and QA history (history). This state management approach improves performance and ensures a coherent user experience across multiple rounds of interaction.

3.7 Application Startup and Public Access

The application startup and public access module is responsible for launching the Streamlit application locally or in a Colab environment and creating a public access link via Cloudflared. It uses subprocess.Popen to start Streamlit in headless mode with CORS and XSRF protection disabled, then starts a Cloudflared tunnel to map the local port to a public HTTPS address, extracting the accessible URL via regex matching. This design allows users to access the application directly without deploying a server, making it convenient for quick sharing in teaching, demonstration, or collaboration scenarios.

4 Experimental Results

The experimental evaluation was conducted to examine the performance and applicability of the proposed RAG framework, which integrates semantic retrieval, hybrid retrieval, and

reranking mechanisms. All experiments were implemented in Google Colab, with deployment provided through a Streamlit interface exposed to external users using Cloudflare tunneling.

The experiments were carried out in a Python 3.11 environment on a GPU-enabled Google Colab instance. The software stack included LangChain, FAISS, BM25, and HuggingFace cross-encoders. Streamlit was selected as the deployment interface, and Cloudflare ensured external accessibility. Two OpenAI models were used: gpt-4o-mini for generation (temperature 0.9 during testing, 0.3 in deployment) and text-embedding-ada-002 (later replaced by text-embedding-3-small for efficiency).

Testing datasets included academic PDFs downloaded via gdown, parsed with PyPDFLoader. Two chunking strategies were tested—recursive character splitting and semantic chunking—both indexed in FAISS and BM25. The evaluation question "What is transformers?" tested semantic and technical retrieval. Hybrid retrieval (BM25 + FAISS) outperformed individual methods, and reranking via BAAI/bge-reranker-base further improved accuracy.

The Streamlit interface allowed users to upload multiple PDFs, switch between "Notes" and "QA" modes, and interactively obtain results in Chinese or English. External users accessed it successfully through Cloudflare, confirming its deployability.

5 Conclusion

This project successfully implements a Streamlit app that combines state-of-the-art information retrieval techniques to answer questions from PDFs. By leveraging FAISS, hybrid retrieval, and reranking, the system provides precise and relevant answers. The integration of OpenAI's chat model enhances response conciseness, offering a practical solution for complex document analysis. Future work may extend to multi-format support and large-scale datasets.

References

1. V. Karpukhin, et al., Dense Retriever: Towards an Efficient and Scalable Retrieval Approach, in Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020).
2. C. Xiong, et al., A Survey of Neural Information Retrieval, *Foundations and Trends® in Information Retrieval* (2021).
3. J. Johnson, et al., FAISS: A Library for Efficient Similarity Search and Clustering of Dense Vectors, Facebook AI Research (2017).
4. Lin Feng, Research on Intelligent Business Data Supply Method Based on LLM and RAG, *China Information Times* (2025), (07): 126–128.
5. Le Gao, Lin Kong, Research on RAG Technology-Driven Intelligent Archive Retrieval, *Zhejiang Archives* (2025-08-22), 1–10.
6. Y. Zhang, Z. Zhang, W. Lai, C. Zhang, T. Gui, Q. Zhang, X. J. Huang, to-tree: Parsing PDF text blocks into a tree, in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Nov. (2024), 10704–10714.
7. P. Gorrell, Syntax and Parsing, Cambridge University Press, Cambridge, UK (1995).
8. S. Husain, P. Gadde, B. Ambati, D. M. Sharma, R. Sangal, A Modular Cascaded Approach to Complete Parsing, in *Proceedings of the 2009 International Conference on Asian Language Processing*, Singapore, Dec. 7–9 (2009), 141–146.

9. K. A. Kiewra, N. F. DuBois, D. Christian, A. McShane, M. Meyerhoffer, D. Roskelley, Note-taking Functions and Techniques, *Journal of Educational Psychology* 83(2) (1991) 240–245.
10. T. Matsubara, Note on the Generating Function of Two-Time Green Functions, *Progress of Theoretical Physics* 32(1) (1964) 50–64.