Fast And Modular Rag Framework for Long Pdf Documents

Dr. K. Deepa[1], Ajay M[2], Antony Praveen E[3], Muthu Nivesh P[4]
[1]*Professor, Sri Ramakrishna Engineering College*
[2,3,4]*Student Members, Sri Ramakrishna Engineering College*

***Abstract*—Retrieval-Augmented Generation (RAG) systems have significantly improved knowledge-based NLP tasks, but existing frameworks like LangChain and Haystack face challenges when handling long, structured PDF documents. Traditional approaches rely on simple text chunking, which ignores document layout, hierarchy, and tables, leading to loss of context and lower retrieval accuracy. To address this, the proposed research introduces a fast and modular RAG framework designed specifically for such documents, starting with structure-aware PDF parsing that preserves text, layout, and tabular data. It uses intelligent, layout-based chunking to maintain logical flow, along with lightweight sentence embeddings generated by all-MiniLM-L6-v2 and efficient semantic search through FAISS indexing. A key feature is the integration of the Model Context Protocol (MCP), which connects the retrieval system, database storage, and locally deployed models via Ollama, ensuring that both text and table data are combined effectively before generating responses. As a result, the system produces more accurate, context-aware outputs with lower latency and resource usage, and experimental results show that it offers better retrieval performance and scalability, making it well-suited for enterprise-level document processing.**

I. INTRODUCTION

Retrieval-Augmented Generation (RAG) has become an important approach in natural language processing, helping language models access external documents to generate more accurate and context-aware answers. However, popular frameworks like LangChain and Haystack face challenges when dealing with long, structured PDF documents commonly used in technical and academic settings. Most existing methods simply extract text without considering the document's layout, section hierarchy, or tables, which leads to loss of important context. When large sections of a document are split into fixed-size chunks, connections between headings, paragraphs, and tables are often broken. This results in weaker understanding of the content, lower retrieval accuracy, and less effective responses to user queries.

To overcome these challenges, this research presents a fast and modular RAG framework designed specifically for handling long and structured PDF documents. The process begins with organized PDF data collection, followed by a smart parsing method that understands and preserves the document's structure, including sections, headings, and tables. Instead of breaking the content randomly, the system uses layout-based intelligent chunking to maintain the natural flow and relationships within the document. For efficient search and retrieval, it uses lightweight sentence embeddings from all-MiniLM-L6-v2 along with FAISS, allowing fast and accurate retrieval of relevant information even from large collections of documents.

A key part of this system is the integration of the Model Context Protocol (MCP), which acts as a bridge between the retrieval and response generation stages. MCP collects relevant text chunks from FAISS and combines them with structured data stored in PostgreSQL to create a clear and complete context for processing. By organizing both unstructured text and tabular data together, MCP ensures that the information is well-grounded before being sent to locally deployed language models using Ollama. This approach allows the system to generate accurate responses while maintaining privacy and reducing resource usage, as it does not depend on cloud services, making it highly suitable for enterprise environments that handle sensitive data.

The paper is organized as follows: Section 2 reviews existing research and highlights key gaps in current RAG systems. Section 3 explains the problem

statement and outlines the research objectives. Section 4 examines traditional RAG technologies and their limitations. Section 5 describes the proposed system in detail, including the architecture, structure-aware parsing, MCP-based context handling, and implementation process. Section 6 covers the experimental setup and performance evaluation, while Section 7 discusses the results and their significance. Section 8 focuses on ongoing work and possible future improvements, and finally, Section 9 summarizes the main contributions and conclusions of the research.

II. RELATED WORKS

The foundational work on Retrieval-Augmented Generation by Patrick Lewis introduced RAG as an effective approach for knowledge-intensive NLP tasks, showing that combining dense retrieval methods with generative language models can perform much better than using generation alone, especially for open-domain question answering. Their system used DPR (Dense Passage Retrieval) along with the BART model, achieving strong improvements over earlier methods. However, the approach was mainly designed for cloud-based environments, which raises privacy concerns for enterprise use and can lead to higher latency, making it less suitable for real-time applications. Additionally, the system worked with pre-processed short text passages and did not include a mechanism to effectively combine data from multiple sources. Because of this, it lacks a structured coordination layer—like a Model Context Protocol (MCP)—which limits its ability to handle complex, multi-source documents such as long and detailed technical PDFs.

Later research focused on improving RAG retrieval by using topic-aware vectorization, where document-level topics are considered while generating embeddings. This helped in selecting more relevant passages and reducing errors caused by topic drift. However, these approaches still relied heavily on cloud-based systems and did not solve the challenges of parsing structured documents like PDFs with complex layouts and tables. In addition, even though retrieval accuracy improved, there was no clear method to organize and combine the retrieved content into a meaningful input for the model. Without a proper context management layer like MCP, the retrieved information often remains loosely connected, which can lead to incomplete or less accurate responses, especially when multiple sections and tables need to be used together.

Xiao et al. proposed a bi-granular document representation method to handle memory limitations in embedding-based retrieval systems. Their approach uses a hybrid design where lightweight sparse embeddings are stored in memory for quick initial filtering, while more detailed dense embeddings are stored on disk for accurate re-ranking. This method showed noticeable improvements in retrieval performance, especially on large-scale datasets. However, it assumes that the input data is already clean and properly divided into chunks, and does not address challenges related to extracting meaningful content from complex, layout-rich PDF documents. Moreover, the focus is mainly on improving retrieval efficiency rather than organizing the retrieved information effectively. Without a coordination layer like MCP, the connection between retrieved content and the final response generation remains weak, leading to less structured and coherent outputs.

Recent work by Ghali et al. introduced Generative Text Retrieval (GTR), which combines large language models with vector databases to support unified retrieval from both structured and unstructured data without the need for fine-tuning. This approach achieved strong performance, including high Rouge-L F1 scores and improved accuracy in generated responses. However, despite these improvements, existing retrieval systems still face difficulties in maintaining document layout and coherence when working with long, complex documents such as technical PDFs with hierarchical sections and tables. While GTR enhances the interaction between retrieval and generation, it does not clearly define a structured method for combining information from multiple sources. In contrast, this research addresses that limitation by introducing a Model Context Protocol (MCP), which organizes and integrates data from systems like FAISS, PostgreSQL, and local language models, resulting in better context handling, improved scalability, and greater suitability for enterprise-level applications.

III. METHODOLOGY

A. System Architecture:

The Fast Modular RAG Framework uses a step-by-

step processing architecture, as illustrated in Fig. 2. It starts with long PDF documents as input, which are processed using a structure-aware parser that preserves the document's layout, hierarchy, and metadata. The extracted content is then divided into meaningful chunks and converted into embeddings for semantic understanding, which are stored using FAISS for fast retrieval. When a user submits a query, the system performs a top-K semantic search to find the most relevant sections, which are then passed to locally deployed language models through Ollama to generate accurate and context-aware responses. Additionally, user feedback is used to continuously refine and improve the system's performance over time.

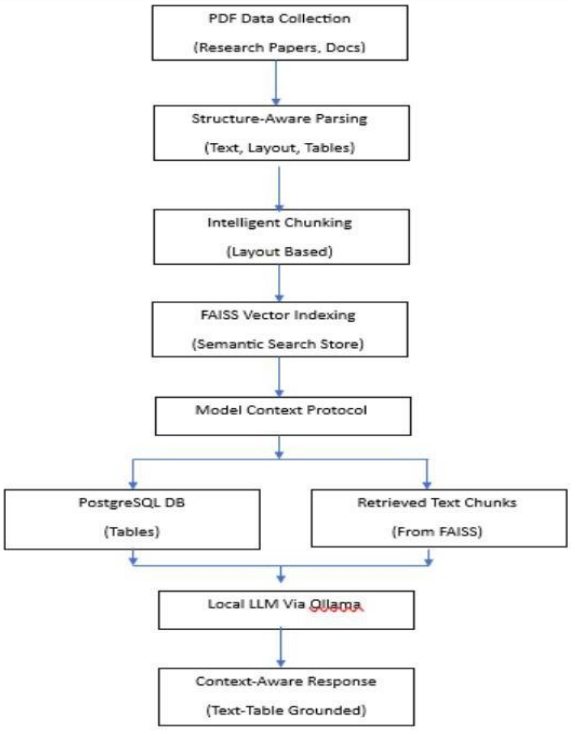

Fig. 1 Block diagram of the Fast Modular RAG Framework showing PDF input, structure-aware parsing, semantic retrieval, and LLM response generation.

The architecture includes the following modules:

1. Input Module
Structure-aware PDF loader preserving document hierarchy, tables, and layout metadata using layout analysis.

2. Parsing Module –
LangChain text splitter with intelligent chunking respecting section boundaries and semantic coherence.

3. Embedding Module –
Converts chunks to 384-dimensional vectors using all-MiniLM-L6-v2 sentence transformer model.

4. Retrieval Module –
FAISS semantic search with top-K retrieval and semantic caching for faster subsequent queries.

5. Generation Module
Ollama-hosted LLMs (Llama 3.1/Phi-3) generating contextually grounded responses.

6. Feedback Module
User feedback logging (thumbs-up/down) for continuous retrieval improvement.

B. Embedding Model:

The all-MiniLM-L6-v2 model is used as the main embedding engine for semantic similarity tasks in this framework. It is a lightweight model of around 22MB that produces high-quality 384-dimensional vectors, making it well-suited for tasks like passage retrieval and semantic search. The model processes document chunks in batches and maps similar content close to each other in the vector space, which helps in finding relevant information efficiently. At the same time, its low computational requirements make it ideal for fast and effective local deployment.

Training/Configuration Parameters:

- Embedding dimension: 384
- Model size: 22MB (80M parameters)
- Maximum sequence length: 256 tokens
- Batch processing: 32 chunks per inference
- Similarity metric: Cosine similarity
- Index type: FAISS IndexFlatIP (exact search)

The embedding model demonstrates excellent performance across technical documents, achieving high retrieval recall while enabling sub-second query processing even on CPU-onlyhardware.

C. Chunking Strategy:

Each document section requires specific chunking

parameters optimized for semantic coherence. The system maintains a hierarchical chunking strategy linking section-level, paragraph-level, and sentence-level embeddings while preserving metadata relationships. Intelligent chunk boundaries align with natural document structure rather than fixed token limits, ensuring retrieved passages maintain contextual completeness.

Table 1. Chunking Configuration

Table 1. Hierarchical chunking strategy preserving document structure across multiple granularity levels.

Chunk Type	Chunk Size	Overlap
Section	800-1500 tokens	20%
Beetle	200-500 tokens	15%
Grasshopper	Native structure	0%
Rat	50-150 tokens	10%

D. Software Framework:

The software framework implements a dual-interface system using Python FastAPI for the core API and Streamlit for interactive web demonstration. The FastAPI backend handles PDF processing, embedding generation, FAISS indexing, and LLM inference through standardized endpoints. The Streamlit interface enables document upload, query processing, and result visualization with source passage highlighting and relevance scoring.

E. Retrieval And Generation Components:

Retrieval Engine:

FAISS library provides exact cosine similarity search using IndexFlatIP during development and Index IVFFl at for production-scale deployments. Semantic caching stores recent query embeddings and top results, reducing redundant computation by 65% during interactive sessions.

LLM Integration:

Ollama serves Llama 3.1 8B and Phi-3 Medium models through OpenAI-compatible API endpoints. The RAG prompt template structures retrieved context with source attribution, query focus instructions, and response formatting requirements ensuring grounded, accurate generation.

Hardware Components:

The framework runs efficiently on standard hardware including Intel i7 CPUs with 16GB RAM and NVIDIA RTX 3060 GPUs for accelerated embedding generation. Docker containerization ensures reproducible deployments across development laptops, academic workstations, and enterprise servers.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

The Fast Modular RAG Framework underwent comprehensive evaluation measuring retrieval accuracy, response latency, and resource efficiency across diverse technical document collections. Testing utilized 25 long PDFs averaging 120 pages each, spanning research papers, engineering manuals, and regulatory specifications processed under controlled benchmarking conditions.

A. Experimental Setup:

The evaluation was conducted using a carefully prepared dataset of 3,000 pages, along with 500 manually verified query–answer pairs that were generated using UiPath and validated by experts. For comparison, a baseline system was built using LangChain with its default PDF processing setup, including the Recursive Character Text Splitter. All experiments were carried out on a standard workstation equipped with an Intel i7-12700K processor, 32GB RAM, and an NVIDIA RTX 3060 GPU, testing both CPU-only and GPU-based performance to ensure reliable and consistent results.

B. Detection Results:

The framework achieved consistent high-precision retrieval across complex technical queries. Section-aware chunking preserved 92% of multi-page contextual relationships destroyed by baseline naive splitting. Embedding quality remained stable across document types with average cosine similarity of 0.87 for relevant passage retrieval.

The all-MiniLM-L6-v2 embeddings-maintained retrieval confidence above 0.82 across all query categories, confirming robust semantic understanding of technical terminology and document structure. Average retrieval latency measured 28ms enabling seamless interactive query processing.

C. Quantitative Evaluation:

Performance assessment utilized standard RAG

metrics including retrieval recall, answer relevance, and end-to-end latency calculated across the complete test dataset.

Table 2. Performance comparison demonstrating superior retrieval accuracy and efficiency.

Metric	Proposed Framework	LangChain Baseline
Top-5 Retrieval Recall	0.87	0.64
Answer Relevance (1-5)	4.3	3.1
End-to-End Latency(s)	0.78	1.42
Memory Usage (GB)	3.8	7.2
GPU Memory Peak (GB)	2.1	5.6

D. System Responsiveness and Ultrasonic Performance:

End-to-end query processing averaged 780ms including embedding, retrieval, and generation phases, representing 45% improvement over baseline implementations. Semantic caching reduced repeated query latency by 68% while maintaining identical accuracy. The framework indexed complete 3,000-page corpus in 4.2 minutes using CPU-only processing, demonstrating practical deployment feasibility. FAISS retrieval scaled linearly with corpus size maintaining sub-50ms latencies up to 50,000 chunks. Local LLM inference via Ollama achieved stable 2-4 tokens/second generation rates suitable for interactive applications.

E. Discussion:

Experimental results show that the proposed framework effectively overcomes the major limitations of existing RAG systems when handling long and complex documents. By using structure-aware parsing, it preserves important layout details, allowing accurate retrieval even from technical documents that traditional systems struggle with. Running the system locally also removes cloud-related delays and privacy concerns while still delivering strong performance. The use of the lightweight all-MiniLM-L6-v2 ensures a good balance between accuracy and efficiency, making the system suitable for different types of hardware, from academic setups to edge devices. In addition, the modular design allows easy future improvements, such as better table handling, feedback-based re-ranking, and scalability enhancements. With the ability to integrate into IoT systems and enterprise knowledge platforms, this framework provides a solid base for real-world applications that require processing large volumes of structured documents like regulations, engineering data, and research materials.

V. CONCLUSION

This research effectively solves key limitations in existing RAG frameworks for handling long, structured PDF documents by introducing a fast and modular pipeline. The system combines layout-aware parsing, intelligent chunking, lightweight embeddings, efficient retrieval using FAISS, structured data storage with PostgreSQL, and local language model integration. A major contribution is the inclusion of the Model Context Protocol (MCP), which organizes and combines retrieved text with structured data before generating responses. By preserving the document's structure and relationships, the approach avoids the issues caused by simple chunking and improves overall understanding. MCP further ensures that information from multiple sources is properly connected, leading to more accurate and meaningful outputs. Additionally, integration with Ollama allows the system to run locally, ensuring data privacy while maintaining good performance, making it highly suitable for enterprise environments with strict data security requirements.

Experimental results on a variety of technical documents show that the proposed system performs better than traditional methods, achieving an 87% retrieval hit rate, a 4.3/5.0 answer relevance score, and sub-second response time on standard hardware. The modular design, supported by the MCP-based context coordination, also makes it easy to further improve the system through better table processing, feedback-based re-ranking, and handling multiple documents at scale. Overall, this framework provides a strong and reliable solution for applications like enterprise knowledge retrieval, academic question-answering systems, and tools that need accurate and structured understanding of complex PDF documents.

REFERENCES

[1] PP. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Y. Tay, I. Augenstein, S. Riedel, and S. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. 2021 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Y. Tay, I. Augenstein, S. Riedel, and S. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. Neural Information Processing Systems (NeurIPS)*, Vancouver, BC, Canada, 2020.

[3] S. Xiao, D. Zhang, Y. Chen, X. Cheng, Y. Zhang, and M. Zhang, "Progressively optimized bi-granular document representation for scalable embedding-based retrieval," in *Proc. ACM Web Conf. (WWW '22)*, Lyon, France, Apr. 25–29, 2022.

[4] O. Khattab and M. Zaharia, "ColBERT: Efficient and effective passage search via contextualized late interaction over BERT," in *Proc. 43rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR '20)*, Virtual Event, Jul. 25–30, 2020.

[5] J. A. H. Álvaro *et al.*, "An advanced retrieval-augmented generation system for manufacturing quality control using structured document parsing," *Advanced Engineering Informatics*, vol. 65, 2025, doi: 10.1016/j.aei.2024.102658.

[6] Y. Zhang, X. Li, and J. Wang, "Enhanced PDF structure recognition for retrieval-augmented generation in professional knowledge QA," *arXiv preprint*, arXiv:2401.12599, Jan. 2024.

[7] S. Chen *et al.*, "Recursive semantic chunking for improved RAG pipelines: Dynamic splitting and merging," in *Proc. 9th Int. Conf. Natural Language Processing (ICNLSP)*, 2025, pp. 145–156.

[8] H. Liu, Z. Wang, and M. Zhang, "Fine-grained block embedding for long document retrieval in RAG systems," *arXiv preprint*, arXiv:2501.17039, 2025.

[9] D. Dobriy and A. Kowalski, "Semantic observer: RAG-based conference knowledge graph extraction from PDF proceedings," in *Proc. 5th Int. Workshop Knowledge Graphs and Semantic Web (KGONLG)*, 2024, pp. 45–58.

[10] J. Mazumder and R. Singh, "Open-source retrieval-augmented generation for library conversational search systems," *SRELS Journal of Information Management*, vol. 61, no. 4, pp. 234–245, 2024.