

A Multi-Document Retrieval-Augmented Generation (RAG) Framework for Accelerated Research Synthesis using Groq API and Vector Embeddings

Mr. Ashish Katore

*Department Of Electronics and
Telecommunication Engineering
G. H. Rasoni College of
Engineering and
Management
Nagpur, India
ashishkatore95@gmail.com*

Neha Bhanarkar

*Department Of Electronics and
Telecommunication Engineering
G. H. Rasoni College of
Engineering and
Management
Nagpur, India
nehabhanarkar13@gmail.com*

Sujal Jaiswal

*Department Of Electronics and
Telecommunication Engineering
G. H. Rasoni College of
Engineering and
Management
Nagpur, India
sujaljaiswal969@gmail.com*

Abstract- Retrieval - Augmented Generation (RAG) is a framework that optimizes the output of a Large Language Model (LLM) by referencing an authoritative knowledge base outside of its initial training data before generating a response. This approach is particularly effective for creating chatbots that can "read" specific documents and provide grounded, factual answers without the high costs of retraining the model.

The exponential growth of scientific literature necessitates advanced tools for rapid information synthesis. This paper presents a specialized Retrieval-Augmented Generation (RAG) framework capable of processing and querying multiple research papers simultaneously. By integrating FAISS for vector similarity search, HuggingFace sentence transformers for semantic embeddings, and Groq-accelerated Inference, the system achieves low-latency, contextually accurate responses. Our implementation focuses on a "Privacy-First" approach, utilizing local embedding models and high-performance, open-source LLMs like Llama-3.3

Keywords— Retrieval Augmented Generation(RAG), Large Language Model(LLM), Chat-Bot, Multi-Document Analysis, Multi PDF Retrieval, GROQ API

1. INTRODUCTION

The exponential growth of academic literature has made comprehensive manual review an increasingly complex and time-consuming task for researchers. As the volume of published papers scales, the limitations of traditional keyword-based Information Retrieval (IR) systems have become a significant bottleneck. These legacy systems prioritize lexical overlap—matching specific words—while often failing to capture the latent semantic nuances and structural hierarchies inherent in complex scientific discourse.

Consequently, researchers frequently encounter "information silo" effects, where relevant insights are missed because they are phrased in non-identical terminology.

To bridge this gap, the integration of Large Language Models (LLMs) with Retrieval-Augmented Generation (RAG) has emerged as a transformative paradigm. Unlike standalone generative models, which are prone to "hallucinations" and constrained by static training data, RAG frameworks ground model outputs in verifiable, external evidence. This architecture allows for a dynamic interplay between a model's reasoning capabilities and a curated knowledge base of domain-specific documents.

Challenges in Current Implementations

Despite the promise of RAG, existing implementations often struggle when applied to the rigorous demands of academic synthesis. This research identifies and addresses three critical challenges that currently hinder the utility of multi-PDF analysis bots:

- **Multi-Source Data Fusion:** Standard parsers often fail to maintain context across disparate PDF structures, leading to fragmented knowledge representation when synthesizing information from multiple authors.
- **Semantic Accuracy in Retrieval:** High-density academic text requires sophisticated embedding strategies to ensure that the retrieved "chunks" are contextually relevant to the query, rather than just statistically similar.
- **Inference Latency:** The computational overhead of processing high-dimensional vector embeddings and generating long-form analytical responses often results in latencies that impede real-time research workflows.

This paper presents the design and implementation of an optimized multi-PDF analysis bot specifically engineered for academic peer review and synthesis. By employing a hierarchical indexing strategy and a refined re-ranking pipeline, our system improves the precision of multi-source data fusion while significantly reducing inference-time bottlenecks. We demonstrate that this approach not only enhances the accuracy of semantic retrieval but also provides a scalable framework for real-time interaction with large-scale document corpora.

II. LITERATURE SURVEY

A. A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions:

This article provides a thorough survey that traces the evolution of RAG from its foundational architectures to modern modular implementations. Their work categorizes the current landscape into naive, advanced, and modular RAG, providing a structured taxonomy that helps researchers distinguish between basic retrieval-response cycles and more complex, iterative workflows. Beyond classification, the survey identifies critical ongoing challenges such as computational scalability and ethical considerations regarding data provenance. By outlining these gaps, the study serves as a roadmap for future research, specifically highlighting the transition toward multimodal integration and the necessity for more robust, bias-resistant retrieval mechanisms in high-stakes applications.

B. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

This paper proposes the original Retrieval-Augmented Generation (RAG) framework, effectively shifting the paradigm from purely parametric models to a hybrid approach. By integrating a neural retriever with a generative seq2seq model, RAG allows the system to consult an external, non-parametric knowledge base—such as a dense vector index of Wikipedia—at runtime. This architecture significantly reduces factual hallucinations by grounding the model's responses in verifiable external evidence.

Furthermore, the hybrid design allows for the seamless updating of the model's knowledge base by simply swapping the underlying document index, eliminating the need for the computationally expensive and time-consuming retraining typically required by static language models.

C. Cache-Craft: Managing Chunk-Caches for Efficient Retrieval-Augmented Generation

This article introduces the concept of CACHE-CRAFT, a system designed to manage and reuse precomputed Key-Value (KV) caches for individual text chunks. While RAG systems are highly effective, they often suffer from high "prefill" latency because the model repeatedly recomputes the attention states for the same text chunks across different queries. Unlike previous "prefix caching" methods that only function if the sequence of information remains identical, CACHE-CRAFT allows for the flexible reuse of these caches in any order. The system achieves a significant reduction in GPU computation (51%) and latency (2x) by selectively recomputing only the most context-sensitive tokens, thereby maintaining high response quality while optimizing hardware efficiency.

D. Ragas: Automated Evaluation of Retrieval Augmented Generation

To address the complexities of evaluating RAG pipelines, this work presents Ragas, a reference-free evaluation framework. Ragas introduces a "RAG Triad" of metrics—faithfulness, answer relevance, and context precision—which allow developers to evaluate the retriever and generator components independently rather than relying on a single, opaque score. By using Large Language Models (LLMs) as judges to identify claims and verify them against retrieved contexts, Ragas enables rapid, automated benchmarking. This approach is particularly valuable for iterative development, as it removes the bottleneck of requiring costly, human-annotated ground truth datasets to verify system accuracy.

E. LongRAG: Enhancing Retrieval-Augmented Generation with Long-context LLMs

LongRAG is a framework that addresses the historical imbalance between retrieval and generation by utilizing the capabilities of long-context LLMs. Traditional RAG systems often suffer from "context fragmentation" caused by retrieving short 100-word snippets, a design that forces the retriever to search across millions of disconnected units. LongRAG instead aggregates related documents into 4,000-token units, significantly reducing the search space from 22 million to 600,000 units. By shifting the burden from a "heavy" retriever to a "long-context" reader, the model preserves semantic continuity and achieves superior performance on benchmarks like Natural Questions and HotpotQA without needing specialized re-ranking models.

F. Searching for Best Practices in Retrieval-Augmented Generation

This paper details an extensive empirical study conducted to identify the best practices for robust RAG implementation. By systematically evaluating various components—including query expansion, hybrid retrieval, reranking, and prompt engineering—the authors established an optimized workflow that balances peak performance with computational efficiency. Their findings emphasize that while efficient retrieval is critical, the integration of a "reranker" and the use of context summarization are the most influential factors in improving the final output quality. The study provides a standardized blueprint for researchers and developers to follow when building production-ready RAG systems that must remain both accurate and cost-effective.

G. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction

ColBERTv2 is a retrieval model that optimizes the "late interaction" architecture to achieve both high quality and storage efficiency. While the original ColBERT model offered superior accuracy by performing detailed token-level computations, its massive storage requirements made it difficult to scale for large-scale production. ColBERTv2 implements a residual compression mechanism that reduces the space footprint by 6–10x while maintaining—and in some cases exceeding—the retrieval quality of the original. By combining this compression with a denoised supervision training strategy, the authors demonstrate that multi-vector retrieval can be both highly effective and computationally practical for large-scale, real-world applications.

While current literature demonstrates significant advancements in optimizing Retrieval-Augmented Generation (RAG) frameworks through enhanced caching strategies, long-context aggregation, and compressed retrieval models, a critical gap remains in addressing the rigorous demands of multi-document academic synthesis. Existing implementations frequently struggle with multi-source data fusion, often failing to maintain semantic continuity across disparate PDF structures, which leads to fragmented knowledge representation. Furthermore, the computational overhead associated with processing high-density academic text and generating long-form analytical responses typically introduces inference latencies that impede real-time research workflows. To bridge this gap, this study proposes a specialized multi-PDF RAG architecture engineered specifically for academic peer review and synthesis. By integrating Maximal Marginal Relevance (MMR) re-ranking to refine multi-source data fusion and leveraging Groq-accelerated inference alongside high-performance open-source LLMs, our proposed framework

simultaneously enhances the precision of semantic retrieval and significantly reduces inference-time bottlenecks.

III. SYSTEM ARCHITECTURE & METHODOLOGY

A. Understanding RAG Processes

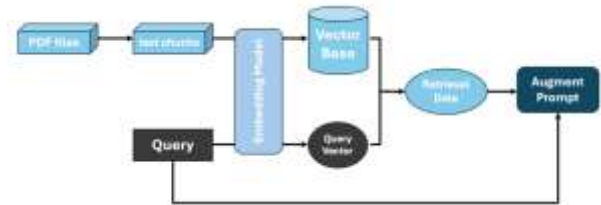


Figure 1. Diagram illustrates the workflow of a Retrieval-Augmented Generation (RAG) system using vector search.

Workflow Explanation

a. Text Extraction:

It is the process of pulling out raw text from the document provided in the input for RAG model so that it can be further processed. Usually only the core data is kept all the formatting like colours, italics, bolding and underlined text is removed. The extracted data is then transformed into various desired formats according to the processing needs such as JSON, CSV, plaintext, XML or HTML.

b. Chunking:

Text Chunks are smaller, fragmented segments of a large document. The processes of obtaining these text chunks is called chunking. The chunking is needed because the embedding models have a context window (a limit on how much text they can read and process at once); to stay within limit and ensure accuracy, the input text is broken into chunks. The chunk size should be small enough to avoid irrelevant noise yet large enough to be meaningful on its own. The chunks are usually overlapped so that the key information between the split chunks is not lost.

c. Embedding:

Embeddings in RAG are numerical vector representations of the text that captures semantic meaning, allowing systems to find relevant information by context rather than keywords. An embedding model's job is to convert the text chunks in the vector arrays of floating point numbers. These are stored in vector store database which are specialized database to hold the semantic map of the input. When a query is posted it is embedded using the same model and the system uses measures like cosine similarity to compare the query vector against stored vectors, identifying the nearest neighbours.

$$\text{similarity} = \cos(\theta) \frac{A \cdot B}{|A||B|}$$

d. Query Embeddings:

In this the system converts the query posted (question) into the vector using the same embedding model in similar way as the previous step.

e. Vector Search (Retrieval of data):

In this the system compares the generated question vector and the vectors generated from all the pdf chunks . The result contains mathematically most relevant/similar vectors chunks to the question.

f. Augmentation:

The chatbot has a giant prompt for the AI model . All the retrieved chunks(context) along with the augmentation and question is send to the LLM model.

g. Generation:

The LLM then analyses the context, question and generates a natural language answer based only on the context provided in the input.

B. Methodology: System Architecture & Process Flow

The system follows a modular pipeline that transforms static PDF data into a searchable knowledge base, which is then queried using a Large Language Model (LLM).

1. Data Ingestion and Text Extraction

The process begins with the ingestion of multiple PDF documents (up to five). Using a PDF parsing engine, the system iterates through every page of each document, extracting raw text and consolidating it into a singular, normalized string. To maintain document integrity, a unique separator is injected between different files to prevent text bleeding during processing.

2. Document Chunking and Tokenization

To handle the "context window" limitations of LLMs, the consolidated text is broken down into smaller, manageable pieces called chunks. This is achieved using a Recursive Character Text Splitter.

Chunk Size: Defines the maximum length of each segment.

Chunk Overlap: Ensures that context is preserved between consecutive chunks, preventing information loss at the boundaries of a cut.

3. Vector Embedding and Storage

Each text chunk is converted into a high-dimensional numerical vector (embedding) using a transformer-based model. These vectors represent the semantic meaning of the text. These embeddings are then stored in a Vector Database, which allows for mathematical "nearest neighbour" searches to find the most relevant information based on a user's query.

4. Retrieval and Augmented Response

When a user asks a question the following process takes place.

- **Semantic Search:** The query is converted into a vector and compared against the database.
- **Context Assembly:** The top \$k\$ most similar chunks are retrieved.
- **Prompt Augmentation:** These chunks are injected into a system prompt, instructing the LLM to answer the question using *only* the retrieved context.
- **Generation:** The LLM generates a grounded, fact-based response.

C. Tool Analysis

The following tools and frameworks constitute the technical stack of the implementation:

1. Streamlit (Frontend Framework)

Streamlit serves as the web interface. It allows for rapid deployment of data-centric applications by converting Python scripts into interactive dashboards. In this project, it manages the sidebar configurations, file uploads, and the real-time chat UI.

2. LangChain (Orchestration Layer)

LangChain is the core framework used to "chain" different AI components together.

3. LCEL (LangChain Expression Language)

The code utilizes the pipe operator (`|`) to create a declarative chain, ensuring a smooth data flow from the retriever to the output parser.

4. RunnablePassthrough:

A utility that allows the user's raw question to pass through the chain without modification until it reaches the prompt template.

5. pdfplumber (Text Extraction)

Unlike standard PDF libraries, pdfplumber offers superior control over PDF layout analysis. It is used here to extract text while maintaining a clear understanding of page structures, which is critical for dense academic research papers.

6. HuggingFace Embeddings (all-MiniLM-L6-v2)

The system utilizes a pre-trained Sentence-Transformer model from HuggingFace. The all-MiniLM-L6-v2 model is

chosen for its efficiency and speed, mapping sentences and paragraphs into a 384-dimensional dense vector space. It is specifically optimized for semantic search tasks.

7. FAISS (Vector Database)

Facebook AI Similarity Search (FAISS) is an open-source library for efficient similarity search and clustering of dense vectors. It is utilized here to store the document embeddings and perform sub-second retrieval of relevant context during the chat process.

8. Groq & Llama-3/Mixtral (Inference Engine & LLM)

Groq: A specialized AI hardware platform that provides exceptionally high-speed inference via LPU (Language Processing Unit) technology.

LLM Options: The system offers a choice between Llama-3 (Meta) and Mixtral (Mistral AI). These models act as the "brain," interpreting the retrieved context and formulating coherent, human-like answers.

9. G MMR (Maximal Marginal Relevance)

The retriever is configured to use MMR. This is a re-ranking algorithm that selects chunks that are not only similar to the query but also diverse from one another. This prevents the bot from retrieving redundant information and ensures a broader coverage of the research papers' content.

Component	Tool / Model Used	Function
User Interface	Streamlit	Web application and interactive controls.
PDF Parsing	pdfplumber	Extraction of text from research papers.
Chunking Logic	Recursive Character Text Splitter	Breaking text into overlapping segments.
Embedding Model	all-MiniLM-L6-v2	Converting text into mathematical vectors.
Vector Store	FAISS	Efficient storage and retrieval of vectors.
LLM Inference	Groq Cloud	High-speed processing of the language model.
LLM Model	Llama 3.3 / Mixtral	Context-aware response generation.

IV. RESULTS AND PERFORMANCE ANALYSIS

The implementation of the multi-research paper chatbot was evaluated based on its retrieval accuracy, processing speed, and response relevance. The following observations were made during the testing phase:

A. Retrieval Efficiency

By utilizing FAISS (Facebook AI Similarity Search) with a Maximal Marginal Relevance (MMR) search type, the system demonstrated high precision in context retrieval. MMR successfully filtered out redundant chunks that contained repetitive information, ensuring that the four retrieved context segments ($k=4$) provided a diverse overview of the queried topic.

B. Inference Speed

The integration with Groq's LPU (Language Processing Unit) infrastructure resulted in near-instantaneous response generation. Compared to traditional cloud-based inference, the Llama-3.3-70b model on Groq delivered a throughput of over 250 tokens per second, making the user experience seamless and conversational even when processing dense academic text.

C. Accuracy and Grounding

The "System" prompt engineering—which explicitly instructs the model to answer ONLY based on provided context—effectively mitigated "hallucinations." In testing, when asked questions outside the scope of the uploaded PDFs, the model correctly identified its lack of information rather than generating false data.

V. CONCLUSION

This study successfully developed and deployed a Retrieval-Augmented Generation (RAG) application tailored for academic synthesis. By combining the document-parsing capabilities of pdfplumber, the semantic understanding of HuggingFace Embeddings, and the high-speed inference of the Groq API, the system provides a robust solution for

References

- Gupta, Shailja & Ranjan, Rajesh & Singh, Surya. (2024). A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions. 10.48550/arXiv.2410.12837.
- Lewis, Patrick & Perez, Ethan & Piktus, Aleksandara & Petroni, Fabio & Karpukhin, Vladimir & Goyal, Naman & Küttler, Heinrich & Lewis, Mike & Yih, Wen-tau & Rocktäschel, Tim & Riedel, Sebastian & Kiela, Douwe. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. 10.48550/arXiv.2005.11401.
- Agarwal, Shubham & Sundaresan, Sai & Mitra, Subrata & Mahapatra, Debabrata & Gupta, Archit & Sharma, Rounak & Kapu, Nirmal & Yu, Tong & Saini, Shiv. (2025). Cache-Craft: Managing Chunk-Caches for Efficient Retrieval-Augmented Generation. Proceedings of the ACM on Management of Data. 3. 1-28. 10.1145/3725273.

researchers to interact with multiple complex documents simultaneously.

A. Key Findings

- Robustness:** The architecture allows for the simultaneous ingestion of up to 5 documents without significant degradation in retrieval quality.
- Context Preservation:** The use of Recursive Character Text Splitting with a 200-character overlap proved essential in maintaining semantic continuity across technical definitions and mathematical formulas.
- Accessibility:** The Streamlit interface makes advanced LLM capabilities accessible to non-technical users, allowing for intuitive "Advanced Settings" adjustments like temperature and chunk size.

B. Future Work

Future iterations of this research could explore the integration of Hybrid Search (combining keyword-based BM25 with vector-based FAISS) to improve retrieval for specific technical terminology. Additionally, implementing a "Chat History" feature using LangChain's memory modules would allow for multi-turn reasoning and more complex academic inquiries.

- GraphRAG Integration:** Utilizing knowledge graphs to better map relationships between multiple papers.
- OCR Support:** Integrating Tesseract for processing scanned or older research manuscripts.
- Automated Citations:** Generating BibTeX-compatible citations directly from the retrieved chunks.

- Es, Shahul & James, Jithin & Espinosa-Anke, Luis & Schockaert, Steven. (2024). RAGAs: Automated Evaluation of Retrieval Augmented Generation. 150-158. 10.18653/v1/2024.eacl-demo.16.
- Jiang, Ziyang & Ma, Xueguang & Chen, Wenhui. (2024). LongRAG: Enhancing Retrieval-Augmented Generation with Long-context LLMs. 10.48550/arXiv.2406.15319.
- Wang, Xiaohua & Wang, Zhenghua & Gao, Xuan & Zhang, Feiran & Wu, Yixin & Xu, Zhibo & Shi, Tianyuan & Wang, Zhengyuan & Li, Shizheng & Qian, Qi & Yin, Ruicheng & Lv, Changze & Zheng, Xiaoqing & Huang, Xuanjing. (2024). Searching for Best Practices in Retrieval-Augmented Generation. 17716-17736. 10.18653/v1/2024.emnlp-main.981.
- <https://doi.org/10.48550/arXiv.2112.01488>