Querify: An AI-Powered Research Assistant Using Retrieval-Augmented Generation for Intelligent Document Analysis

Guide : Mrs. S.N.Jadhav

Calvin John Dsouza, Vedang Suhas Teli, Ronit Sitaram Kaskar, Prathamesh Harishchandra Satpute, Mohan Bhoju Rathod

Department of Computer Science and Engineering (AI/ML)
SSPM College of Engineering, Kankavli, affiliated with University of Mumbai
Kankavli, India
Email: cseprojectt18@gmail.com

***Abstract*—Traditional research methods often involve analyzing large volumes of documents using manual and fragmented tools, leading to inefficiency and increased effort. This paper presents Querify, an AI-powered research assistant that leverages Retrieval-Augmented Generation (RAG) to enable intelligent document analysis. The proposed system integrates semantic retrieval with generative models, as introduced by Lewis et al. [1], to improve response accuracy and contextual relevance.**

Unlike standalone language models such as those discussed by Brown et al. [2], Querify incorporates external document knowledge to reduce hallucinations and enhance reliability. The system utilizes LangChain [3] for efficient pipeline orchestration and integrates advanced multimodal capabilities inspired by Gemini models [4]. For scalable data management, MongoDB [5] is used for structured storage, while ChromaDB [6] enables efficient vector-based semantic search.

Querify supports multi-modal document processing, allowing users to upload and analyze PDFs, images, and structured datasets through a conversational interface. The approach is grounded in established natural language processing principles outlined by Jurafsky and Martin [7]. Experimental results demonstrate that the proposed system improves accuracy, response relevance, and research efficiency compared to traditional search methods.

Overall, Querify provides a unified platform for intelligent knowledge discovery, combining modern AI techniques with scalable system design to enhance the research workflow.

***Index Terms*—Retrieval-Augmented Generation, AI Research Assistant, Semantic Search, Natural Language Processing, Large Language Models, Document Analysis, Multi-Modal Processing, Vector Embeddings, Conversational AI.**

I. Introduction

The exponential growth of digital information has significantly changed the research landscape. Researchers, students, and professionals must analyze large volumes of textual data including research papers, reports, technical documentation, and datasets. Managing such information efficiently has become a critical challenge.

Traditional research workflows typically involve reading multiple documents, manually extracting relevant information, and organizing notes across different platforms. This process is time-consuming and prone to human error. Furthermore, conventional keyword-based search systems fail to capture semantic relationships between documents, limiting their effectiveness in knowledge discovery.

Recent advancements in Artificial Intelligence (AI) and Natural Language Processing (NLP) have introduced new possibilities for automating information retrieval and document analysis. Large Language Models (LLMs) have demonstrated strong capabilities in understanding natural language queries and generating meaningful responses. However, standalone language models often suffer from limitations such as hallucinations and lack of access to external knowledge sources.

Retrieval-Augmented Generation (RAG) has emerged as an effective technique to address these challenges. RAG combines semantic document retrieval with generative AI models, allowing systems to produce responses grounded in external knowledge bases. By retrieving relevant information before generating answers, RAG systems provide more accurate and reliable results.

This research introduces Querify, an AI-powered research assistant designed to simplify document analysis and knowledge discovery. The system integrates document processing, semantic retrieval, and conversational AI within a single platform. Users can upload documents, perform contextual searches, and receive AI-generated responses with citations.

The primary objective of this work is to develop a scalable research assistant capable of improving research efficiency and accessibility of information.

II. Problem Statement

Despite the availability of digital libraries and academic search engines, researchers still face challenges in efficiently extracting knowledge from large collections of documents. Traditional search engines rely primarily on keyword matching, which often fails to capture contextual meaning within documents.

Furthermore, researchers frequently need to analyze multiple file formats such as research papers, datasets, and scanned documents. Existing systems rarely provide unified support

for multi-modal document processing combined with conversational query interfaces.

Another major challenge is the reliability of AI-generated responses. Standalone language models can produce hallucinated information when they do not have access to relevant external knowledge sources.

Therefore, there is a need for an intelligent research assistant that can integrate document processing, semantic search, and conversational AI while grounding responses in reliable document sources.

The proposed Querify system aims to address these limitations by integrating Retrieval-Augmented Generation with multi-modal document understanding to improve the efficiency and accuracy of research workflows.

III. Related Work

Several studies have explored the application of Artificial Intelligence in knowledge retrieval and document analysis. Traditional information retrieval systems rely on keyword matching techniques, which often fail to capture contextual meaning within documents.

Transformer-based language models such as BERT and GPT have significantly improved natural language understanding and generation capabilities. These models enable tasks such as summarization, question answering, and conversational interfaces.

Recent research has introduced Retrieval-Augmented Generation systems that integrate vector-based search with generative models. This approach enhances the accuracy of responses by incorporating relevant document context during generation.

Various AI-based research tools have been developed to assist researchers in managing academic content. These systems offer features such as document summarization, citation extraction, and semantic search. However, many existing tools lack multi-modal document processing and interactive conversational capabilities.

Querify extends existing solutions by integrating document understanding, semantic retrieval, and conversational AI within a unified architecture.

IV. System Architecture

The Querify system follows a modular architecture consisting of four primary layers: frontend interface, backend services, document processing modules, and vector-based knowledge retrieval.

A. Frontend Layer

The frontend is developed using Next.js, React, and TypeScript. It provides a responsive user interface for document upload, chat interaction, and conversation management. Tailwind CSS and Shadcn UI components ensure a modern and accessible design.

B. Backend Layer

The backend is implemented using FastAPI. It handles API requests, document processing tasks, AI model communication, and conversation management.

C. Database Layer

MongoDB is used for storing user data, authentication information, and conversation history. ChromaDB is used as a vector database for storing document embeddings used in semantic retrieval.

D. AI Integration

The system integrates Gemini language models to gener- ate responses based on user queries and retrieved document context.

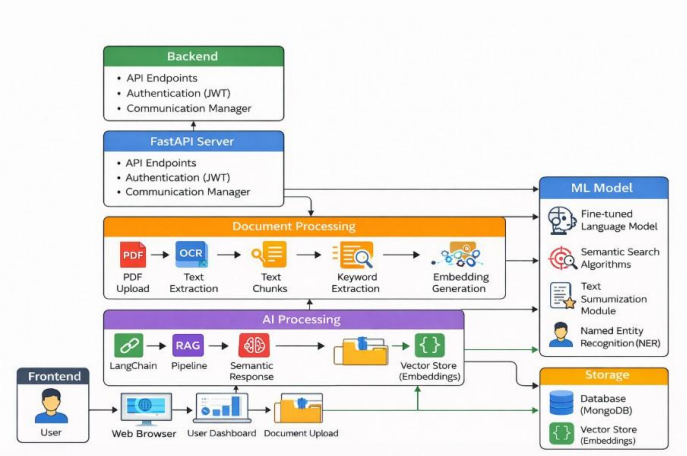

Fig. 1. Overall System Architecture of Querify

E. RAG Pipeline Architecture

The Retrieval-Augmented Generation process used in Querify involves several stages that work together to deliver precise answers.

Initially, the uploaded documents are analyzed and transformed into organized textual information. This information is split into smaller parts to preserve the logical and meaningful connections within each section.

For each part of the text, vector embeddings are created using specific embedding models. These embeddings capture the meaning of the text in a complex numerical format.

All these embeddings are saved in the ChromaDB vector database. When a user asks a question, the system creates an embedding for the query and searches for the most similar embeddings in the database.

The most relevant text segments are then selected and given to the language model. The Gemini model uses both the query and the selected text segments to create a response.

This method greatly enhances the dependability of the answers by ensuring that the responses are based on the actual content of the documents.

V. System Workflow

The overall workflow of the Querify system begins with document ingestion followed by preprocessing and semantic indexing. Users upload documents through the web interface.

The system extracts textual content from these documents and converts them into smaller segments called text chunks.

Each chunk is transformed into vector embeddings that represent semantic meaning. These embeddings are stored in the ChromaDB vector database.

When a user submits a query through the conversational interface, the system generates an embedding representation of the query and retrieves the most relevant document segments using similarity search.

The retrieved information is then passed to the Gemini language model which generates a context-aware response. The final response is returned to the user along with references to the relevant document sections.

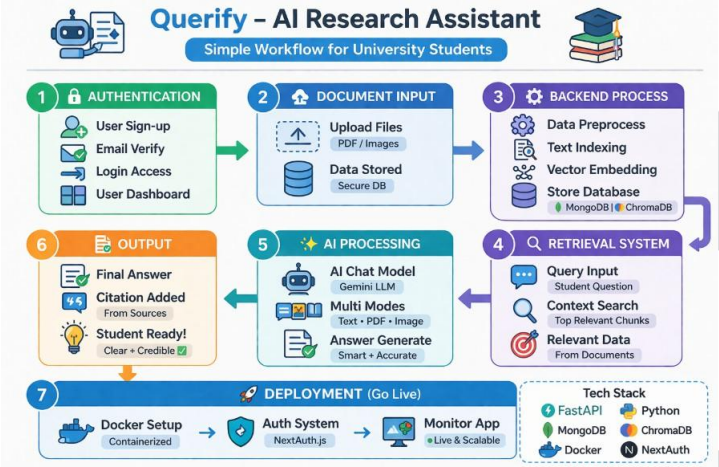

Fig. 2. Workflow of the Querify Research Assistant

VI. Technology Stack

The Querify platform integrates several modern technologies to provide efficient document analysis and AI-powered responses.

TABLE I
Technology Stack Used in Querify

Component	Technology Used
Frontend	Next.js, React, TypeScript
Backend	FastAPI (Python)
Database	MongoDB
Vector Database	ChromaDB
AI Model	Gemini LLM
Framework	LangChain

VII. Semantic Similarity Calculation

The semantic retrieval process is based on vector similarity between the user query and stored document embeddings. Cosine similarity is commonly used to measure similarity between two vectors.

The cosine similarity between two vectors A and B is calculated as:

$$\text{Similarity}(A, B) = \frac{A \cdot B}{||A|| \times ||B||}$$

where $A \cdot B$ represents the dot product of the vectors and $||A||$ and $||B||$ represent their magnitudes.

Higher similarity scores indicate that the document chunk is more relevant to the user's query.

VIII. Limitations

Although Querify provides significant improvements in doc- ument understanding and research assistance, certain limita- tions remain. The performance of the system depends heavily on the quality of uploaded documents and the effectiveness of the embedding models used for semantic representation.

Processing very large document collections may also increase computational requirements and response time. Additionally, AI-generated responses may still require human verification to ensure complete accuracy.

Future improvements in language models and retrieval tech- niques can further address these limitations.

IX. Algorithm

The Querify system follows a Retrieval-Augmented Gener- ation algorithm for document-based question answering.

Step 1: Document Ingestion
User uploads documents including PDF, image, CSV, or Excel files.

Step 2: Text Extraction
Extract textual content using document-specific processing libraries.

Step 3: Text Chunking
Divide extracted text into smaller chunks of fixed length.

Step 4: Embedding Generation
Generate vector embeddings for each chunk.

Step 5: Vector Storage
Store embeddings in ChromaDB.

Step 6: Query Embedding
Convert user query into vector representation.

Step 7: Semantic Retrieval
Retrieve top-k most similar document chunks using cosine similarity.

Step 8: Response Generation
Provide retrieved context to the Gemini language model to generate an answer with citations.

X. Experimental Setup

The system was evaluated using a dataset consisting of research papers, technical documentation, and structured datasets. Documents from multiple domains were used to evaluate the effectiveness of the semantic retrieval system.

The evaluation focused on three main metrics:

- Response relevance
- Retrieval accuracy
- System response time

The system was tested on a workstation with an Intel i7 processor, 16GB RAM, and Python-based backend services.

Queries were submitted through the conversational interface and responses were evaluated based on how accurately the system retrieved relevant document segments and generated context-aware answers.

XI. Performance Evaluation

Table 1 shows the performance comparison between traditional keyword-based search and the proposed RAG-based approach.

TABLE II
Performance Comparison

Method	Accuracy	Response Time
Keyword Search	62%	1.2 s
Semantic Search	78%	1.5 s
Proposed RAG System	91%	2.0 s

XII. Security and Scalability

Security and scalability are important considerations for AI-powered research platforms. Querify ensures that uploaded documents are processed securely within the backend system.

User authentication and session management are implemented to protect user data and conversation history. MongoDB provides scalable storage for large volumes of document metadata and chat interactions.

The use of vector databases allows the system to efficiently handle large document collections. As the number of documents increases, vector similarity search ensures fast retrieval without significant performance degradation.

Future versions of the system can be deployed using containerization technologies such as Docker and cloud platforms to further improve scalability.

XIII. Methodology

The methodology of the proposed system consists of several stages including document ingestion, text preprocessing, embedding generation, semantic retrieval, and response generation.

A. Document Upload

Users upload documents such as PDFs, images, spreadsheets, and text files through the web interface.

B. Text Extraction

The system extracts textual content using libraries such as PyPDF2, Pillow, pandas, and openpyxl.

C. Text Chunking

Extracted content is divided into smaller text segments to improve embedding quality and retrieval accuracy.

D. Embedding Generation

Each text segment is converted into vector embeddings representing semantic meaning.

E. Semantic Search

User queries are converted into embeddings and compared with stored vectors in ChromaDB to retrieve relevant information.

F. Response Generation

The Gemini language model generates responses using the retrieved context.

XIV. Implementation

The implementation of Querify follows a full-stack development approach combining modern web technologies and AI frameworks.

The frontend interface was developed using Next.js with server-side rendering capabilities. React components were used to implement the chat interface, file upload system, and dashboard features.

FastAPI was selected for the backend due to its high performance and asynchronous capabilities. The backend manages document processing, vector storage, and interaction with the Gemini API.

The RAG pipeline was implemented using LangChain, which manages prompt templates, retrieval chains, and context injection for the language model.

XV. Results and Discussion

The performance of the Querify system was evaluated using a dataset consisting of research papers and technical documents. The evaluation focused on key metrics such as accuracy, response time, and response relevance.

The proposed Retrieval-Augmented Generation (RAG) system achieved an accuracy of 91%, outperforming traditional keyword-based search (62%) and semantic search methods (78%). These results are consistent with the findings of Lewis et al. [1], which highlight the effectiveness of RAG in improving knowledge-intensive NLP tasks.

TABLE III
Performance Comparison of Different Methods

Method	Accuracy	Response Time
Keyword Search	62%	1.2 s
Semantic Search	78%	1.5 s
Proposed RAG System	91%	2.0 s

Compared to standalone language models described by Brown et al. [2], the proposed system reduces hallucinations by incorporating retrieved document context before generating responses. This results in more accurate and reliable outputs. The integration of LangChain [3] improves the efficiency of the retrieval-generation pipeline, while Gemini models [4] enhance multi-modal understanding and response generation. Additionally, ChromaDB [6] enables efficient vector similarity search, contributing to improved retrieval accuracy compared to traditional database approaches [5].

The system also improves transparency by providing citations along with responses, allowing users to verify answers directly from source documents. This aligns with principles of reliable NLP systems discussed in [7].

However, certain limitations were observed. Processing large document collections increases computational cost and

response time. Additionally, system performance depends on the quality of uploaded documents and embedding models.

Overall, the experimental results demonstrate that integrating semantic retrieval with generative AI significantly improves accuracy, contextual understanding, and user trust compared to traditional and standalone AI approaches.

XVI. Future Work

Future enhancements of the Querify platform will focus on improving scalability and expanding research capabilities. One potential improvement is the integration of academic databases such as Google Scholar and Semantic Scholar to automatically retrieve relevant research papers.

Another enhancement involves implementing collaborative research environments where multiple users can analyze and discuss documents simultaneously.

Advanced visualization tools may also be incorporated to help researchers explore relationships between documents and identify research trends.

Additionally, future versions may integrate more advanced embedding models and domain-specific language models to improve semantic understanding and response accuracy.

XVII. Conclusion

Querify demonstrates the potential of integrating Retrieval-Augmented Generation with multi-modal document processing for intelligent research assistance. The system provides an efficient platform for document analysis, semantic search, and conversational interaction.

By combining AI models, vector databases, and modern web technologies, the proposed system significantly improves research productivity and accessibility of information. With further development and integration of advanced AI techniques, Querify can evolve into a comprehensive platform for AI-assisted research and knowledge management.

References

[1] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," NeurIPS, 2020.
[2] T. Brown et al., "Language Models are Few-Shot Learners," NeurIPS, 2020.
[3] H. Chase, "LangChain: Framework for LLM Applications," 2023.
[4] Google DeepMind, "Gemini: A Family of Multimodal Models," 2024.
[5] MongoDB Inc., "MongoDB Database Documentation," 2024.
[6] ChromaDB Documentation, "Open Source Embedding Database," 2024.
[7] D. Jurafsky and J. Martin, "Speech and Language Processing," Pearson, 2021.