IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

SmartDocs AI – Smart Way to Search Your Document

Vyankatesh Rajejadhav, Smit Lawande, Suyash Sonawane, Shruti Dalvi, Prof. Swati Powar
Student, Computer Science, MIT ADT University, Pune, India
Professor, Computer Science Engineering, MIT ADT University, Pune, India

Abstract: *In today's information-driven world, extracting useful knowledge from large volumes of unstructured documents such as PDFs is a time-consuming task. This paper presents SmartDocs AI, an intelligent document interaction system that allows users to converse with their PDF data using natural language. The system is built on the Retrieval-Augmented Generation (RAG) framework, which combines information retrieval with generative artificial intelligence. The proposed model extracts text from uploaded PDF files, divides it into smaller chunks, and generates vector embeddings to capture semantic meaning. These embeddings are stored in a vector database for efficient similarity search. When a user asks a question, the system retrieves the most relevant text segments and provides a precise, context-aware answer using a Large Language Model (LLM). This approach improves factual accuracy, reduces hallucination, and enables users to gain meaningful insights directly from their documents. The application is developed using a modular architecture with React.js for the front end, Flask for the backend, and MinIO for secure file storage. Future enhancements include integrating Optical Character Recognition (OCR) for scanned PDFs and extending the model to handle image-based data for broader multimodal document understanding..*

Keywords: Retrieval-Augmented Generation (RAG), Large Language Model (LLM), PDF Chatbot, Generative AI, Natural Language Processing (NLP), Information Retrieval, Text Embedding, Flask, React, MinIO, Document Intelligence, Semantic Search, OCR Integration, Smart Query System

I. INTRODUCTION

In the present era of information explosion, organizations and individuals generate massive volumes of digital documents every day in the form of reports, research papers, manuals, and e-books. Locating specific information within these unstructured data sources is often labor-intensive and time-consuming. Conventional keyword-based search engines retrieve only lexical matches and fail to capture the semantic context of user queries. Hence, there is an increasing demand for intelligent document-search systems that can understand natural language and return precise, context-aware answers.

SmartDocs AI addresses this challenge by integrating Retrieval-Augmented Generation (RAG) with modern Large Language Models (LLMs) to create an interactive document-search assistant. Instead of merely returning matching sentences, the system extracts text from uploaded PDF documents, converts it into vector embeddings that represent semantic meaning, and stores them in a vector database. When a user asks a question, SmartDocs AI retrieves the most relevant text chunks and generates a coherent answer grounded in the source content. This approach reduces hallucination, enhances factual accuracy, and significantly improves user experience compared with traditional search techniques.

The system is implemented using a modular web architecture comprising a React.js front end for user interaction, a Flask back end for processing and retrieval, and MinIO object storage for secure document management. SmartDocs AI demonstrates how combining information-retrieval methods with generative AI can revolutionize document exploration in domains such as research, education, and corporate knowledge management.

DOI: 10.48175/IJARSCT-33710

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

II. LITERATURE REVIEW

The evolution of document retrieval and intelligent question-answering has undergone significant transformation over the past two decades. This section reviews the key milestones that have led to the development of RAG-based document interaction systems such as SmartDocs AI.

A. Information Retrieval and Document Understanding

Earlier systems for document understanding mainly relied on keyword-based search and rule-based text extraction. These methods were limited because they could not understand the context or semantic meaning behind user queries. With advancements in Natural Language Processing (NLP), models such as TF-IDF (Term Frequency–Inverse Document Frequency) and BM25 were introduced to rank and retrieve relevant text sections. However, these methods still struggled with semantic understanding, working only when the user's query exactly matched the document's keywords [1].

B. Evolution of Large Language Models (LLMs)

The introduction of Large Language Models such as GPT (Generative Pretrained Transformer), BERT (Bidirectional Encoder Representations from Transformers), and T5 brought a major shift in how machines understand human language. These models are trained on massive text datasets and can generate human-like responses. They form the core of interactive systems like ChatGPT, which can understand, reason, and respond naturally to user questions. However, standalone LLMs have a fundamental limitation—they do not have direct access to private or domain-specific data such as user-uploaded PDF files. This limitation led to the development of Retrieval-Augmented Generation (RAG) systems [3].

C. Retrieval-Augmented Generation (RAG)

RAG is a modern technique that combines information retrieval with generative AI. It retrieves relevant information from external sources such as PDFs and provides it to the LLM for answer generation. In a typical RAG workflow, a document is uploaded and converted into text; the text is split into smaller segments called chunks; these chunks are transformed into embeddings (numerical representations of semantic meaning); when a user poses a question, the system identifies the most relevant chunks based on embedding similarity; and finally these chunks are passed to the LLM, which generates an accurate, context-aware response. Studies have shown that RAG improves factual accuracy, reduces hallucination, and makes answers more reliable by grounding each response in actual document data [3][4].

D. Embeddings and Vector Databases

Embeddings are at the core of RAG-based systems. Models such as OpenAI Embeddings and Sentence-BERT convert text into dense numerical vectors that encode semantic meaning [7]. Vector databases such as FAISS, Pinecone, ChromaDB, and Weaviate are then used to store and compare these embeddings efficiently [6][8]. This approach allows the system to understand meaning rather than just keywords, making it possible to retrieve accurate results even when the phrasing of the question differs from the wording in the document [1][2].

E. System Implementation Technologies

Modern document AI systems rely on web-based frameworks for real-time interaction and scalability. React.js provides a user-friendly interface for uploading documents and conducting chat sessions. Flask handles data processing, embedding generation, and communication with the LLM API. MinIO or equivalent cloud storage platforms store files securely and enable easy retrieval. Such architectures make the system fast, efficient, and accessible to users across domains including research, education, and business [5].

F. Future Directions

The future of document-based AI systems lies in expanding beyond plain text to multimodal understanding. Integrating OCR will allow the system to read and interpret scanned PDFs or image-containing documents. Future versions can also process tables, graphs, and figures, making the system more powerful for research and industrial use cases. Additionally, fine-tuning domain-specific LLMs will improve accuracy for specialized fields such as legal, medical, or academic document analysis.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

III. METHODOLOGY

The methodology for developing SmartDocs AI using Retrieval-Augmented Generation (RAG) is organized into multiple well-defined phases. Each phase ensures that the system performs efficiently, provides accurate answers, and delivers a smooth user experience.

A. Data Collection and Preprocessing

The system works primarily with PDF documents uploaded by users. These documents may include research papers, reports, technical manuals, or other text-rich materials. Before processing, the system performs the following preprocessing steps:

Text Extraction: The PDF is converted into raw text using libraries such as PyMuPDF or pdfminer.

Text Cleaning: Unnecessary characters, extra whitespace, and irregular line breaks are removed to ensure clean, consistent text.

Chunking: The cleaned text is divided into smaller, meaningful segments (chunks) for better contextual understanding and retrieval.

Metadata Handling: Each chunk is tagged with metadata such as page number or section title to preserve context for accurate response generation.

This preprocessing stage ensures that the document is in a format suitable for embedding generation and semantic retrieval.

B. Embedding Generation

After preprocessing, each text chunk is converted into an embedding—a dense numerical vector that captures the semantic meaning of the segment. Embeddings are generated using pre-trained models such as OpenAI Embeddings or Sentence-BERT. These representations enable the system to understand semantic similarity, allowing it to match user queries to the most relevant document segments even when exact wording differs. All embeddings are stored in a vector database to facilitate fast semantic search rather than simple keyword matching, thereby improving retrieval accuracy.

C. Retrieval-Augmented Generation (RAG) Workflow

The RAG model forms the core of SmartDocs AI. It combines the strengths of information retrieval and generative AI to produce accurate, context-grounded responses. The workflow proceeds as follows:

User Query: The user uploads a PDF document and enters a natural language question.

Similarity Search: The system converts the query into an embedding and searches the vector database to find the most semantically relevant chunks.

Context Preparation: The retrieved chunks are combined to form a coherent context that provides the factual basis for answering the question.

Answer Generation: The context is passed to a Large Language Model (e.g., GPT or equivalent), which generates a clear, fluent, natural-language answer grounded in the document content.

This workflow ensures that all system responses are anchored in the actual document rather than relying solely on the LLM's parametric knowledge, thereby reducing hallucination.

D. System Architecture and Integration

The project is built using a client–server architecture to ensure smooth communication between the user interface and the backend processing units. The frontend (React.js) provides a user-friendly web interface for uploading PDFs, entering queries, and displaying responses in a conversational chat format. The backend (Flask) handles text extraction, chunking, embedding creation, vector retrieval, and communication with the LLM API. The storage layer (MinIO) provides secure object storage for uploaded PDF files and processed data. This modular design ensures scalability, ease of debugging, and smooth data flow between all components.

E. Testing and Evaluation

The system undergoes multiple testing stages to ensure reliability and performance. Unit testing verifies each component—PDF extraction, chunking, embedding, retrieval, and response generation—individually. Integration

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 12, April 2026
Impact Factor: 8.2

testing ensures smooth communication between the frontend, backend, and database layers. Performance testing evaluates response latency and retrieval accuracy across documents of varying sizes. User acceptance testing assesses overall usability to confirm that users can easily interact with the system and obtain relevant, accurate answers.

IV. SYSTEM DESIGN

The system design of SmartDocs AI is structured using a modular, scalable architecture to ensure seamless interaction between users and their documents through natural language. The system follows a client–server model, integrating multiple technologies for data processing, storage, and intelligent query handling. The design emphasizes efficiency, accuracy, scalability, and security while maintaining a user-friendly interface.

A. System Architecture Overview

The overall architecture is divided into three primary layers: (1) Frontend Layer (User Interface), (2) Backend Layer (Processing and Intelligence), and (3) Storage Layer (Data and File Management). Each layer performs specialized tasks and communicates through RESTful APIs to ensure modularity and smooth data exchange.

B. Architectural Components

The Frontend (React.js) provides an interactive, web-based interface that allows users to upload PDF documents, enter natural language queries, and view responses in a conversational chat format. It implements real-time updates using asynchronous API calls to the backend.

The Backend (Flask Framework) acts as the core processing unit of the system. It handles text extraction from uploaded PDFs using libraries like PyMuPDF or pdfminer, cleans and structures the extracted text, generates numerical embeddings using models such as OpenAI Embeddings or Sentence-BERT, performs similarity search within the vector database to find the most relevant chunks, and uses the LLM to produce accurate, context-aware answers grounded in document data.

The Storage Layer consists of MinIO for secure object storage of uploaded PDFs and processed data, and a vector database (FAISS, Pinecone, or ChromaDB) for storing and retrieving embeddings through efficient semantic search. This combination ensures fast retrieval and maintains scalability for large document sets.

The LLM Integration connects the system to a Large Language Model via API (e.g., GPT-based model). The model generates human-like responses by utilizing retrieved context from the vector database, ensuring that answers are factually grounded and coherent.

C. Workflow of the System

The SmartDocs AI system operates through the following sequential workflow. (1) Document Upload: The user uploads a PDF through the web interface and the file is securely stored in MinIO. (2) Text Extraction and Preprocessing: The backend extracts text from the PDF, cleans it, divides it into logical chunks, and preserves metadata such as page number and section title. (3) Embedding Generation: Each chunk is converted into a vector embedding and stored in the vector database for semantic similarity search. (4) User Query Processing: The user enters a natural language question; the system generates an embedding for the query and performs similarity search to retrieve the most relevant chunks. (5) Contextual Answer Generation: Retrieved chunks are passed as context to the LLM, which generates a coherent, context-grounded answer. (6) Response Display: The frontend renders the generated response in a chat-style format, enabling continuous multi-turn interaction with the same document.

V. RESULTS

The SmartDocs AI system was deployed locally and evaluated using multiple PDF documents of varying sizes and domains, including research papers, policy documents, and technical manuals. The system demonstrated the ability to accurately answer natural language questions by retrieving semantically relevant text segments from the uploaded documents.

In qualitative testing, the system successfully responded to factual queries with precise, context-grounded answers. For example, when asked “What is the national education policy?” after uploading a relevant government document, the

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 12, April 2026 Impact Factor: 8.2

system correctly retrieved and summarized the policy framework without hallucination. The chat-style interface allowed users to ask follow-up questions within the same session, maintaining document context across multiple turns.

The modular architecture ensured stable performance across document sizes ranging from a few pages to multi-hundred-page reports. Response times remained acceptable for interactive use, and the system correctly attributed answers to specific page-level contexts through the metadata-aware chunking approach. Future quantitative evaluation will include precision, recall, and ROUGE metrics to provide a comprehensive performance benchmark.

VI. CONCLUSION

This paper presented SmartDocs AI, an intelligent document interaction system that enables users to query large PDF documents using natural language. By integrating the Retrieval-Augmented Generation (RAG) framework with a Large Language Model, the system overcomes the limitations of traditional keyword-based search by providing semantically accurate, context-aware answers grounded in uploaded document content.

The modular architecture—comprising a React.js frontend, Flask backend, MinIO storage, and a vector database—ensures scalability, ease of maintenance, and smooth real-time interaction. The RAG approach significantly reduces hallucination and improves factual accuracy compared to standalone LLM-based systems.

Future work will focus on integrating OCR capabilities to support scanned and image-based PDFs, extending the system toward multimodal document understanding (tables, graphs, and figures), and fine-tuning domain-specific LLMs for specialized use cases in legal, medical, and academic domains. SmartDocs AI represents a significant step toward making static digital archives fully interactive and conversationally accessible.

ACKNOWLEDGMENT

The authors express their profound gratitude to their guide, Prof. Swati Powar, for her expert guidance, encouragement, and inspiration throughout this project work.

The authors would like to thank Prof. Dr. Vipul Dalal, Director, Department of Computer Science and Engineering, for extending all support during the execution of the project.

Sincere thanks are also due to Prof. Dr. Shradha Phansalkar, Head, Department of Computer Science and Engineering, MIT School of Engineering, MIT-ADT University, Pune, for providing the necessary facilities to complete this work.

The authors are grateful to Prof. Dr. Rajneeshkaur Sachdeo, Dean, MIT School of Engineering, MIT-ADT University, Pune, for providing the infrastructure to carry out this project. The authors also thank all faculty members of the Department for their invaluable support and advice.

REFERENCES

[1]. Weaviate, "Vector Embeddings Explained," [Online]. Available: https://weaviate.io/blog/vector-embeddings-explained

[2]. Weaviate, "Vector Search Explained," [Online]. Available: https://weaviate.io/blog/vector-search-explained

[3]. P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4]. OpenAI, "Retrieval-Augmented Generation (RAG) Concepts and Implementation," OpenAI Documentation. [Online]. Available: https://platform.openai.com/docs

[5]. LangChain, "LangChain Documentation," [Online]. Available: https://python.langchain.com

[6]. Pinecone, "Pinecone Documentation," [Online]. Available: https://docs.pinecone.io

[7]. Hugging Face, "Transformers Documentation," [Online]. Available: https://huggingface.co/docs/transformers

[8]. ChromaDB, "ChromaDB Documentation," [Online]. Available: https://docs.trychroma.com

[9]. Neon, "PostgreSQL Tutorial," [Online]. Available: https://neon.com/postgresql/tutorial

[10]. DataCamp, "pgvector Tutorial," [Online]. Available: https://www.datacamp.com/tutorial/pgvector-tutorial