Global Journal of Engineering Innovations & Interdisciplinary Research

Original Article

Multi-PDF RAG Chatbot Using LangChain and Streamlit

G Murali[1], U Venu Gopal, A Jithendra Reddy, S Hemalatha

[1]Assistant Professor, Department of Computer Science and Engineering, JNTUA College of Engineering, Pulivendula
[2]M.Tech Students, Department of Computer Science and Engineering, JNTUA College of Engineering, Pulivendula

Correspondence

Dr. G. Murali, M.E, Ph.D.

Assistant Professor, Department of Computer Science and Engineering, JNTUA College of Engineering, Pulivendula

- Received Date: 25 May 2025
- Accepted Date: 15 June 2025
- Publication Date: 27 June 2025

Abstract

The exponential growth of unstructured textual data, particularly in portable document format (PDF), presents significant challenges in extracting, summarizing, and retrieving actionable knowledge. This research presents an intelligent, lightweight, and scalable web-based application—Multiple PDF Streamlit—that bridges traditional document handling with cutting-edge AI capabilities. Powered by Large Language Models (LLMs) and enhanced through Retrieval-Augmented Generation (RAG), the system enables seamless ingestion, parsing, and semantic interrogation of multiple PDF documents in parallel. By employing a hybrid architecture that combines text chunking, embedding-based vector search, and context-aware generation, the platform offers dynamic question-answering, multi-document summarization, and an interactive user interface for knowledge exploration. The backend pipeline leverages modern frameworks like LangChain and FAISS/Chroma for efficient retrieval, while the front-end is built using Streamlit, providing a real-time, user-friendly interface. This synthesis of NLP, semantic search, and interactive AI creates an end-to-end system capable of transforming static PDFs into a living, searchable knowledge base. The application not only democratizes access to LLM-powered insights but also exemplifies the future of explainable and interactive document intelligence systems

Keywords

Streamlit; PDF Document Processing; Natural Language Processing (NLP); Large Language Models (LLM); Retrieval-Augmented Generation (RAG); Multi-PDF Summarization; Semantic Search; Embedding-based Retrieval; Contextual Question Answering; Vector Databases; LangChain; Document Ingestion Pipeline; Text Chunking; Information Retrieval; Generative AI; Interactive Web Application; User-centric NLP Interface; AI-powered Knowledge Base; End-to-End LLM Pipeline; Explainable Document Analysis.

Introduction

Large Language Models (LLMs) are great at generating responses that sound human, but there's a catch — they're limited by the data they were trained on. In fast-changing fields where new information constantly emerges, these models can quickly become outdated. This can lead to incomplete answers or even inaccuracies, especially when the stakes are high and decisions rely on up-to-date knowledge.

That's where Retrieval-Augmented Generation (RAG) comes in. RAG enhances LLMs by allowing them to pull information from external sources — like PDFs, databases, or websites — in real time. This means the model isn't just guessing based on past training data; it's using current, verified information to generate responses. That makes RAG an ideal approach for tasks that require deep, accurate knowledge.

In this report, we walk through our experience building a RAG system that uses PDF documents as its main source of knowledge. We share our process step-by-step, from design choices to development to evaluation. Along the way, we highlight the technical challenges we faced and how we tackled them. We also compare working with commercial tools like OpenAI's models and open-source alternatives such as LLaMA, especially when it comes to data privacy and security.

Our goal is to provide practical guidance for anyone looking to build or improve a RAG-based system — whether you're a developer, researcher, or organization — and help you make smart choices around accuracy, reliability, and transparency based on your specific needs.

Literature Survey

This section delves into prior work that lays the foundation for our approach, highlighting ethical concerns, the evolution of document processing, the transformative role of large language models (LLMs), and the mechanics behind Retrieval-Augmented Generation (RAG) systems.

Ethical Perspectives and the Evolution of Document Processing

Recent studies have compared traditional scenario- based chatbots with those powered by large language models (LLMs), focusing

Citation: Murali G, Venu Gopal U, Jithendra Reddy A, Hemalatha S. Multi-PDF RAG Chatbot Using LangChain and Streamlit. GJEIIR. 2025;5(5):098

on how each handles recommendations. Beyond their technical differences, the analysis raised key ethical questions around fairness, transparency, privacy, and accountability in LLM-driven systems. The findings emphasized the need to embed ethical thinking into the development and evaluation of conversational AI, and called for stronger industry practices and deeper academic research to ensure responsible AI deployment.

In parallel, the field of document processing has undergone a major shift, especially with the growth of NLP technologies. LLMs now play a critical role in enhancing chatbot capabilities, particularly in dealing with the overwhelming volume of textual data. Personalized chatbots that can summarize documents and answer user queries have become increasingly important.

One notable approach involved using extractive summarizers to pull out core insights from scientific papers, helping address information overload. Researchers found that considering the structural complexity of text led to more effective summaries—especially for languages that lack robust NLP toolsets. This underlined the value of deeper text comprehension in generating high-quality, concise summaries. The study also explored how conversational interfaces, like chatbots, intersect with AI ethics in practical applications.

The Rise of Large Language Models (LLMs)

The emergence of LLMs such as GPT-3.5, GPT-4, LLaMA, and Mistral has significantly reshaped natural language processing. These models have shown exceptional performance in tasks like summarization, translation, code generation, and open-ended question answering. Despite their strengths, they still face limitations in multi-document scenarios, constrained by fixed context windows, memory challenges, and occasional factual inaccuracies.

One study introduced LangChain, a framework designed to streamline the process of querying information from PDF documents using LLMs. By combining natural language processing with user- friendly tools like Streamlit, LangChain simplified data access and improved the efficiency of retrieving relevant information from dense text sources.

Another line of research explored the idea of using LLMs themselves as reference standards—or "oracles"—for evaluating summary quality. Methods like GPTScore and GPTRank, supported by contrastive learning techniques, allowed smaller summarization models to perform on par with larger LLMs when judged using LLM-generated criteria. Experiments on benchmark datasets such as CNN/DailyMail and XSum confirmed that this approach could make smaller models more effective, reducing the need for heavy computational resources while maintaining quality.

UnderstandingRetrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) blends two key components of NLP: Information Retrieval (IR) and Natural Language Generation (NLG). First proposed by Lewis et al., RAG enhances language model responses by retrieving relevant information from large external datasets before generating the output. This significantly improves both the accuracy and contextual relevance of responses.

Unlike traditional language models that rely solely on their internal training data (often outdated or incomplete), RAG systems dynamically fetch up-to- date knowledge from external sources. This grounding process ensures that the generated content reflects the most current and factual information available.

A typical RAG workflow consists of several core stages:

Data Collection

The process starts with gathering domain-specific text data from various sources such as PDFs, structured files, or plain text documents. This curated collection serves as the foundation for a custom knowledge base, allowing the system to respond with more targeted and accurate answers.

An illustration of this workflow can be seen in Figure 1, showing how RAG systems enhance the capabilities of LLMs by anchoring outputs in real-time, relevant data.

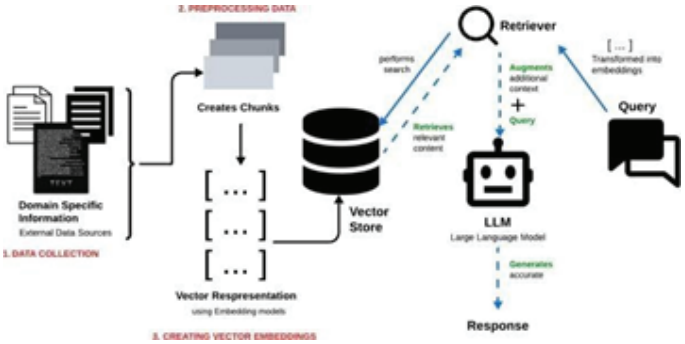

Fig. 1: Architecture of Retrieval Augmented Generation(RAG) system.

Data Preprocessing

Once the raw data is collected, it goes through a preprocessing stage to make it cleaner and more usable. This involves removing unnecessary elements like special characters or formatting issues, standardizing the text, and then breaking it down into manageable parts—usually smaller chunks or tokens such as words or phrases. Segmenting the text this way is crucial because it helps the system later on by making the retrieval process faster and more accurate.

Creating Vector Embeddings

After the data is cleaned and chunked, each segment is converted into a numerical format—known as a vector embedding. Embedding models like BERT or Sentence Transformers are used here to capture the meaning and context of each chunk in a high- dimensional vector form. These vectors allow the system to measure semantic similarity between a query and stored data. All these vectors are then saved in a Vector Store—a special type of database designed for fast and efficient similarity searches.

Retrieving Relevant Information

When a user enters a query, it's first converted into a vector embedding using the same method applied to the documents. The Retriever component of the system then compares this query vector with those stored in the Vector Store, pulling out the most relevant data chunks. This ensures that the information feeding into the next step is both highly relevant and up-to-date.

Augmenting Context

Next, the retrieved information is blended with the general

knowledge already built into the language model. This dual approach combines the LLM's built- in understanding with current, domain-specific content from external documents. By doing this, the system grounds its output in both long-term knowledge and real-time data, creating a more well- rounded and context-aware response.

Response Generation by the LLM

The user's original question—now enriched with the most relevant supporting information—is passed to a Large Language Model like GPT, T5, or LLaMA. The model uses this full, context-packed input to generate a response that is not only fluent and natural- sounding, but also factually accurate and directly tied to the source material.

Final Output

Unlike traditional language models that may produce vague or inaccurate answers, RAG systems are designed to be transparent and reliable. By tying their responses back to real data, they help reduce hallucinations, improve accuracy, and make it easier to trace where the information came from. The result is a more trustworthy, precise, and insightful AI- generated response.

Modular NLP Pipelines: LangChain and Beyond

Modern frameworks like LangChain, Haystack, and LlamaIndex have made it significantly easier to build Retrieval-Augmented Generation (RAG) systems. These toolkits take care of much of the behind-the- scenes complexity—like loading documents, chunking text, generating embeddings, setting up vector stores, and chaining components together. They also support multi-step reasoning, integration with APIs, and customized prompt design, making them powerful tools for creating intelligent document agents.

Low-Code Web Deployment with Streamlit

As AI systems become more advanced, there's a growing need to make them easy to interact with. Streamlit has become a go-to platform for this because it allows developers to build interactive web apps with minimal coding. Unlike traditional front-end frameworks, Streamlit is fast, responsive, and doesn't require deep knowledge of web development. Its seamless compatibility with NLP tools and RAG workflows has enabled the creation of real-time, user- friendly research applications—perfect for demos, prototypes, or lightweight production tools.

Innovation Behind the Multiple PDF Streamlit Project

The Multiple PDF Streamlit Project brings all of these technologies together into a practical, user-friendly application. It provides a complete pipeline that can:

- Load and parse multiple PDF documents,
- Break content into context-aware chunks and embed them semantically,
- Retrieve the most relevant information using vector search,
- Use LLMs to summarize content and answer user questions,
- And deliver all of this through a clean, interactive Streamlit interface.

This project is a powerful example of how cutting- edge tools like LLMs and RAG can be brought into real-world use through accessible, low-code environments. It opens up advanced AI features to users who may not have a technical background, blending automation with thoughtful user interface design.

Contributions to Research and Industry

This project pushes the boundaries in several key areas: multi-document question answering, semantic exploration of documents, and making AI more transparent and explainable. Its modular, easy-to- adapt design makes it ideal for deployment in sectors like healthcare, legal, education, and enterprise knowledge management.

Not only does it serve academic research with its focus on explainability and semantic reasoning, but it also provides a practical framework for real-world document intelligence. By combining real-time feedback, transparent pipelines, and user-centered design, this system aligns well with the goals of Explainable AI (XAI) and next-generation NLP applications.

Framework Architecture

The core goal of this project is to develop a web-based application that allows users to upload PDF documents and receive intelligent summaries and responses powered by large language models. This system integrates tools such as Streamlit for the frontend interface, LangChain for NLP pipeline management, and OpenAI's APIs for text generation and embeddings.

OpenAI is a leading research organization that focuses on advancing artificial intelligence technologies for societal benefit. They conduct extensive research across domains such as natural language processing, robotics, and reinforcement learning. Among their most impactful contributions are the Generative Pre- trained Transformer (GPT) models, which can produce human-like text from a wide range of inputs. These models support tasks like summarization, translation, and natural language understanding, and they serve as the backbone of this system.

The system's architecture, as illustrated in Figure 2, begins with the user uploading a PDF document. The content is then extracted using the PdfReader class from the PyPDF2 Python library, which is widely used for parsing and reading PDF files. Once the document text is extracted, it is broken into smaller, manageable segments known as chunks. These chunks are necessary for efficient processing and retrieval.Each chunk is converted into a numerical representation known as an embedding, which captures its semantic meaning. These embeddings are essential for tasks such as similarity search and semantic comparison. The embeddings used in this project are generated using the OpenAI Embeddings class, accessed via the LangChain library. This allows us to represent each chunk of text in a high-dimensional space where similar meanings are placed closer together

The embeddings are then stored in a vector database, which serves as a knowledge base for semantic search. This database allows for rapid retrieval of relevant

content based on the user's query. When a user submits a question, the system first converts it into an embedding and then performs a semantic search through the vector store. The most relevant chunks of information are retrieved and ranked based on similarity.

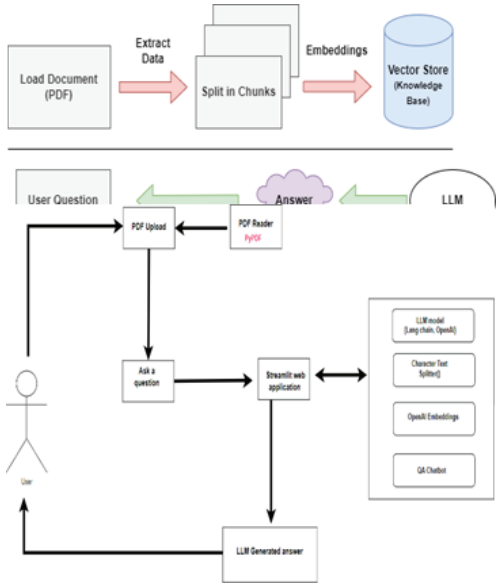

Figure 2. Architecture of the Model

Figure 3. Proposed Block Diagram

Results and Implementation

A confidential API key was generated through the OpenAI platform and securely stored in a local environment, ensuring it is treated with the same level of security as a password to prevent public exposure. Once the API key was generated (as shown in Figure 4), it enabled the integration of OpenAI's services into our project for research and application development.

The frontend of our application was built using Streamlit, a user-friendly open-source Python tool that simplifies the development of web interfaces. It allows developers to design interactive and responsive web applications directly from Python

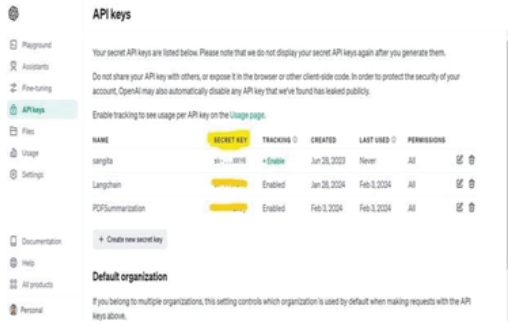

Figure 4. Creating the OpenAI API Key

Figure 5. Interface of the Application

scripts without requiring knowledge of frontend technologies like HTML, CSS, or JavaScript. With minimal code, users can add interactive widgets such as sliders, buttons, and data input fields [18].

As shown in Figure 5, our application's interface displays informational content on the left side of the screen, with the core feature — a chatbot for querying PDF files — as the main focus. This solution integrates Streamlit with LangChain and OpenAI's language models, providing a quick and efficient way for users to get answers from uploaded PDFs.

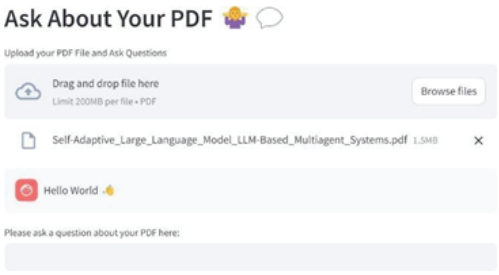

Figure 6. Uploading the PDF

The system begins by verifying if a PDF file has been uploaded (i.e., checking whether the 'pdf' variable is not null). Once a file is detected, a PdfReader object is created using the PyPDF2 library to extract its contents (as illustrated in Figure 6). The script iterates through the file, extracting text from each page and compiling it into a single string for processing.

Upon successful upload of the PDF, the question- answering chatbot is initialized. The user can then pose queries based on the content of the PDF (Figure 7 shows an example).

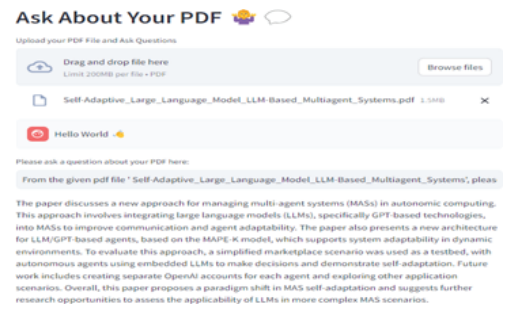

Figure 7. Responses Generated by the Model Results

Prompt 1

Adaptive_Large_Language_Model_LLM- Based_ Multiagent_Systems', please summarize the pdf in approximately 200 words including all the details that have been discussed in the paper.

The paper introduces an innovative strategy for managing multi-agent systems (MASs) within autonomic computing. It focuses on leveraging large language models (LLMs), specifically GPT-based models, to enhance the adaptability and communication among agents. The study presents a new architectural framework based on the MAPE-K model, facilitating dynamic adaptation of systems. The proposed model was tested using a marketplace simulation, where autonomous agents demonstrated decision-making and adaptive behaviors using embedded LLMs. Future work aims to assign individual OpenAI accounts to each agent and explore additional real-world scenarios. The findings suggest a paradigm shift in how MASs can become more self- adaptive through LLM integration, opening up new research avenues for complex agent-based systems.

Prompt 2: What is the Self-adaptive Large Language Model?

Result:

A self-adaptive large language model (LLM) refers to an AI framework integrated into multi-agent systems (MASs) to improve their communication and flexibility. These models, such as GPT-4, enable agents to adjust their behaviors dynamically and handle intricate scenarios effectively. This adaptability is rooted in the MAPE-K loop, a system management framework that supports decision- making in fluctuating environments (refer to Figure 9).

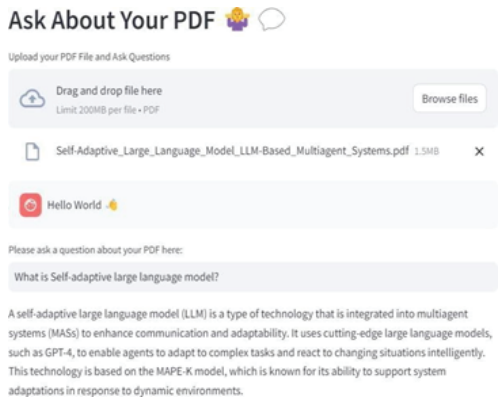

Figure 8*. Results for Prompt 2*

The original study explored a self-adaptive MAS powered by LLM/GPT-based architecture, emphasizing autonomous reasoning and intelligent decision-making capabilities in agents.

Our experiments evaluated the system using a variety of documents across different domains. The system excelled in extracting critical information from PDFs and generating precise summaries, outperforming conventional methods. The chatbot demonstrated quick and accurate responses within seconds, ensuring user satisfaction. The retrieval process first fetches relevant documents, and a QA chain then constructs responses from the retrieved content. The system used the 'stuff' chain type in which multiple documents are appended to the prompt for processing by the LLM. Semantic search significantly enhanced the relevance and accuracy of results by interpreting user intent and context effectively.

Preliminary Evaluation of the Guide

Feedback Process Overview

This report underwent an informal evaluation aimed at gathering feedback for the section: "Using OpenAI's Assistant API: GPT Series" [4.2.1]. Although the feedback process was informal, it provided valuable insights that contributed to refining the guide. The feedback received from participants indicated that the workshop was successful. The majority of attendees were able to follow the guide effectively and successfully implemented their RAG models by the end of the session.

2	anonymous	Well structured of instructions but copy code should be with terminal command
3	anonymous	Copy-pasting code from the pdf files.
4	anonymous	Some code problems...
5	anonymous	it is very easy to understand
6	anonymous	Havent face any challenges
7	anonymous	Editor and environment issues not relating to the guide.

(b) Primary Area of Expertise

Figure 9: Demographic Information from Participants

The feedback was gathered from a small but diverse group during the workshop. A total of eight individuals completed a demographics form, which helped us understand the backgrounds and technical expertise of the participants. The group comprised individuals with varying levels of experience in machine learning, natural language processing (NLP), and using tools for Retrieval Augmented Generation (RAG). Most participants were familiar with Python and OpenAI models.

KeyFeedbackPoints

Throughout the session, participants shared their thoughts on how their understanding of RAG systems had improved, which aspects of the workshop they found most valuable, challenges they encountered, and suggestions for future improvements. Below are the key points highlighted by the participants.

Before attending the workshop, most participants reported a reasonable level of familiarity with RAG systems. This background allowed for more in-depth discussions during the session. After the workshop, a noticeable improvement in participants' understanding of RAG systems was observed.

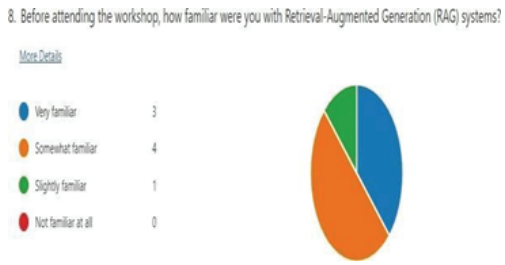

Figure 10: Participants' Familiarity with RAG Systems

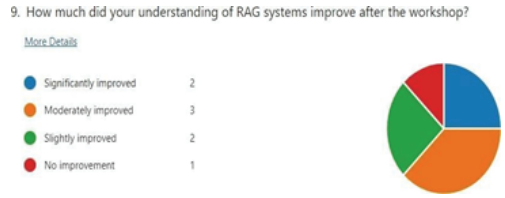

Figure 11: Participants' Improvement in Understanding RAG Systems

The majority of participants highlighted the practical coding exercises as the most valuable aspect of the workshop, which helped them better understand how to implement RAG systems. Additionally, several participants emphasized the importance of the discussions that followed the exercises.

Figure 12: Most Valuable Aspects of the Workshop

Incorporating Feedback to Improve the Guide

The evaluation also provided insight into how the guide could be improved, particularly in terms of clarifying instructions and streamlining the implementation process. Technical issues were the most commonly raised concerns, particularly those related to errors that occurred when copying from PDF files. To address this, error handling was incorporated into the code snippets to throw meaningful errors, allowing users to run the code smoothly.

2	anonymous	Well structured of instructions but copy code should be with terminal command
3	anonymous	Copy-pasting code from the pdf files.
4	anonymous	Some code problems...
5	anonymous	it is very easy to understand
6	anonymous	Havent face any challenges
7	anonymous	Editor and environment issues not relating to the guide.

Figure 13: Feedback on Challenges Faced During the Implementation of the Guide

Several participants shared that while the guide was helpful, there were occasional challenges related to the technical setup, particularly the handling of errors when dealing with complex PDF files.

In conclusion, the feedback gathered during the evaluation was invaluable in confirming the effectiveness of the approach outlined in the guide. By testing it in a hands-on workshop environment and discussing the improvements in the RAG models, we were able to address areas where practitioners faced difficulties. The guide's iterative improvements, based on real-world feedback, not only made it more user- friendly but also demonstrated the importance of continuous enhancement driven by user input.

12. Any other comments or suggestions for future improvement?

3 Responses

ID ↑	Name	Responses
1	anonymous	My last remark is about the vectoreStore, I just found out it is stored in OpenAI's server, so there should be a warning to not upload any sensitive documents. And also, I want to know , in the guide: where it is stored, and how to delete it.

Figure 14: Comments and Suggestions for Improving the Guide

Discussion

Professionals in sectors such as healthcare, legal services, and customer support frequently face challenges when working with static models that depend on outdated or narrowly scoped information. Retrieval-Augmented Generation (RAG) models present a practical alternative by drawing real-time insights from relevant sources. The transparent, traceable decision-making process enabled by RAG models enhances trust, especially in domains where evidence-based conclusions are vital.

In this study, a RAG implementation guide was developed and evaluated through a workshop where participants followed structured steps to build and deploy RAG systems. This hands-on approach provided a valuable, practice-oriented contribution, equipping users with clear, executable instructions to integrate RAG into their workflows. In doing so, the guide contributes to the expanding library of AI- powered problem-solving tools.

Moreover, RAG technology introduces promising research possibilities poised to influence the future landscape of natural language processing (NLP) and artificial intelligence (AI). As the ecosystem matures, several areas of growth emerge, including optimized information retrieval strategies, dynamic data adaptation, and support for diverse data formats like visual and audio content. The rapid advancement of supportive tools has further propelled the deployment of RAG systems. Key emerging trends in this domain include:

1. **Haystack:** This open-source platform combines dense and sparse retrieval techniques with advanced language models. It supports real-time search applications and can be used to build RAG solutions for tasks such as summarization, document retrieval, and question answering [4].
2. **Elasticsearch with Vector Search:** Modern enhancements to Elasticsearch allow it to handle dense

vector searches efficiently. When integrated with systems like Faiss, Elasticsearch facilitates hybrid retrieval strategies, effectively balancing precision and performance for large-scale datasets [3].

3. **Integration with Knowledge Graphs:** Researchers are actively investigating the use of structured data sources like knowledge graphs to enrich RAG models. This approach is aimed at boosting the factual consistency and reasoning depth of the models, making them more dependable for data-intensive use cases [8].
4. **Adaptive Learning and Continual Fine- Tuning:** A rising focus area is enabling RAG systems to incrementally improve using new inputs and user interaction. These techniques help maintain model relevance and accuracy in constantly changing information environments [7].
5. **Cross-Lingual and Multimodal Capabilities**: Future developments are expected to extend RAG systems across multiple languages and data types. Incorporating capabilities for multilingual retrieval and processing of non-textual data (e.g., images or audio) will significantly broaden the scope and impact of RAG systems [2]. Future research will likely prioritize adaptability, multilingual support, and deeper integration with heterogeneous data to tackle increasingly sophisticated information needs.

Limitations

While the proposed framework offers substantial benefits, it does have certain constraints. It depends on pre-trained large language models, such as OpenAI's GPT, which may struggle to accurately interpret or summarize complex and highly specialized content outside their original training distribution. Although these models generate fluent and coherent responses, their effectiveness diminishes with niche or uncommon topics.

Moreover, the reliance on cloud-based solutions such as OpenAI introduces potential concerns related to data security, user privacy, and long-term platform availability. These limitations underscore the importance of model fine-tuning and careful deployment planning to maintain the reliability and resilience of chat-based applications, especially in domains requiring high accuracy and sensitivity.

Conclusion & Future Recommendations

This study presents a comprehensive methodology for building tailored chatbot systems leveraging large language models (LLMs), with a focus on tasks like question answering and document summarization. By incorporating tools like Streamlit, LangChain, and OpenAI's models, the framework effectively mitigates the challenge of information overload, enabling users to extract valuable insights from dense textual data.

The guide serves as a practical tutorial, showing developers how to construct full-scale applications for summarization and question-answering using modern AI stacks. The integration of powerful LLMs with LangChain's NLP features and Streamlit's accessible interface design results in a flexible, efficient solution ideal for researchers and developers tackling complex text processing tasks.

For future development, several enhancements are recommended:

- Model fine-tuning to increase domain specificity and reliability
- Integration of adaptive AI to improve model responsiveness to new data
- Expansion of chatbot functionality, including broader task support and cross- domain capabilities

This framework holds the potential to transform user interaction with textual data by improving efficiency, fostering knowledge discovery, and increasing overall productivity across various fields.

Lastly, it is essential to reiterate the challenges associated with dependence on external APIs like OpenAI. Data governance, long-term access, and system robustness remain critical concerns. Thoughtful planning, including the possibility of local deployment or hybrid models, will be crucial to ensure the safe and sustainable use of such advanced technologies.

References

1. Balage Filho, Pedro Paulo, TA Salgueiro Pardo, and M. das Gracas Volpe Nunes. "Summarizing scientific texts: Experiments with extractive summarizers." In Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007), pp. 520-524. IEEE, 2007.
2. Bang, Junseong, Byung-Tak Lee, and Pangun Park. "Examination of Ethical Principles for LLM-Based Recommendations in Conversational AI." In 2023 International Conference on Platform Technology and Service (PlatCon), pp. 109-113. IEEE, 2023.
3. Prasad, Rajesh S., U. V. Kulkarni, and Jayashree R. Prasad. "Machine learning in evolving connectionist text summarizer." In 2009 3rd International Conference on Anti- counterfeiting, Security, and Identification in Communication, pp. 539-543. IEEE, 2009.
4. Nalini, N., Agrim Narayan, Akshay Mambakkam Sridharan, and Arkon Pradhan. "Automated Text Summarizer Using Google Pegasus." In 2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES), pp. 1-4. IEEE, 2023.
5. Patil, Dinesh D., Dhanraj R. Dhotre, Gopal S. Gawande, Dipali S. Mate, Mayura V. Shelke, and Tejaswini S. Bhoye. "Transformative trends in generative ai: Harnessing large language models for natural language understanding and generation." International Journal of Intelligent Systems and Applications in Engineering 12, no. 4s (2024): 309-319.
6. Topsakal, Oguzhan, and Tahir Cetin Akinci. "Creating large language model applications utilizing langchain: A primer on developing llm apps fast." In International Conference on Applied Engineering and Natural Sciences, vol. 1, no. 1,pp. 1050-1056. 2023.
7. Monks, Thomas, and Alison Harper. "Improving the usability of open health service delivery simulation models using Python and web apps." NIHR Open Research 3 (2023).
8. Pokhrel, Sangita, and Shiv Raj Banjade. "AI Content Generation Technology based on Open AI Language Model." Journal of Artificial Intelligence and Capsule Networks 5, no. 4 (2023): 534-548.
9. S, Adith Sreeram A, and Pappuri Jithendra Sai. "An Effective Query System Using LLMS and Langchain." International Journal of Engineering Research & Technology,July 4, 12(6), 2023. 367 -369

10. Liu, Yixin, Alexander R. Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. "On learning to summarize with large language models as references." arXiv preprint arXiv:2305.14239 (2023).
11. Prasad, Rajesh S., U. V. Kulkarni, and Jayashree R. Prasad. "Machine learning in evolving connectionist text summarizer." In 2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication, pp. 539-543. IEEE, 2009.
12. Nalini, N., Agrim Narayan, Akshay Mambakkam Sridharan, and Arkon Pradhan. "Automated Text Summarizer Using Google Pegasus." In 2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES), pp. 1-4. IEEE, 2023.
13. Patil, Dinesh D., Dhanraj R. Dhotre, Gopal S. Gawande, Dipali S. Mate, Mayura V. Shelke, and Tejaswini S . Bhoye. "Transformative trends in generative ai: Harnessing large language models for natural language understanding and generation." International Journal of Intelligent Systems and Applications in Engineering 12, no. 4s (2024): 309-319.
14. Topsakal, Oguzhan, and Tahir Cetin Akinci. "Creating large language model applications utilizing langchain: A primer on developing llm apps fast." In International Conference on Applied Engineering and Natural Sciences, vol. 1, no. 1, pp. 1050-1056. 2023.
15. Monks, Thomas, and Alison Harper. "Improving the usability of open health service delivery simulation models using Python and web apps." NIHR Open Research 3 (2023).
16. Pokhrel, Sangita, and Shiv Raj Banjade. "AI Content Generation Technology based on Open AI Language Model." Journal of Artificial Intelligence and Capsule Networks 5, no. 4 (2023):534-548.
17. S, Adith Sreeram A, and Pappuri Jithendra Sai. "An Effective Query System Using LLMS and Langchain." International Journal of Engineering Research & Technology,July 4, 12(6), 2023. 367 -369
18. Liu, Yixin, Alexander R. Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. "On learning to summarize with large language models as references." arXiv preprint arXiv:2305.14239 (2023).
19. Prasad, Rajesh S., U. V. Kulkarni, and Jayashree R. Prasad. "Machine learning in evolving connectionist text summarizer." In 2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication,pp. 539-543. IEEE, 2009.
20. Nalini, N., Agrim Narayan, Akshay Mambakkam Sridharan, and Arkon Pradhan. "Automated Text Summarizer Using Google Pegasus." In 2023 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES), pp. 1-4. IEEE, 2023.
21. Patil, Dinesh D., Dhanraj R. Dhotre, Gopal S. Gawande, Dipali S. Mate, Mayura V. Shelke, and Tejaswini S.Bhoye. "Transformative trends in generative ai: Harnessing large language models for natural language understanding and generation." International Journal of Intelligent Systems and Applications in Engineering 12, no. 4s (2024): 309-319.
22. Topsakal, Oguzhan, and Tahir Cetin Akinci. "Creating large language model applications utilizing langchain: A primer on developing llm apps fast." In International Conference on Applied Engineering and Natural Sciences, vol. 1, no. 1, pp. 1050-1056. 2023.
23. Monks, Thomas, and Alison Harper. "Improving the usability of open health service delivery simulation models using Python and web apps." NIHR Open Research 3 (2023).
24. Pokhrel, Sangita, and Shiv Raj Banjade. "AI Content Generation Technology based on Open AI Language Model." Journal of Artificial Intelligence and Capsule Networks 5, no. 4 (2023):534-548.
25. S, Adith Sreeram A, and Pappuri Jithendra Sai. "An Effective Query System Using LLMS and Langchain." International Journal of Engineering Research & Technology,July 4, 12(6), 2023. 367 -369
26. Liu, Yixin, Alexander R. Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. "On learning to summarize with large language models as references."