Journal of IoT in Social, Mobile, Analytics, and Cloud (ISSN: 2582-1369)
www.irojournals.com/iroismac/

AI-Driven RAG Chatbot: Combining Information Retrieval with Generative AI

Venkatesh S.[1], Dhanya K R.[2], Kaniska P.[3]

[1-3]Computer Science with Data Analytics, Dr N.G.P Arts and Science College, Coimbatore, India

E-mail: [1]venkateshkaviya1234@gmail.com, [2]dhanyafab@gmail.com, [3]kaniskaprabhakaran573@gmail.com

Abstract

Generative AI technologies are emerging nowadays and they transform the way of user interaction with information, and allows the systems to deliver accurate responses to the user queries. This research focuses on creating a Retrieval Augmented Generation Chatbot as an e-learning assistant where it fetches the accurate data from the pdf document that is trained on and give accurate precise responses to the user query. This e-learning assistant is created specifically for the subject of "Artificial Intelligence" to deliver the user-queries related to Artificial Intelligence. The system uses Flask for the backend and React for the frontend. PDFs are loaded, split into smaller sections, and processed using LangChain. Embeddings are generated with Google's AI models and stored in Chroma, a vector database. When a user submits a query, the system searches for similar content and uses Google Gemini-1.5-Pro to generate a response based on the retrieved data. This ensures high accuracy by relying on specific content rather than broad AI knowledge. This solution can easily scale and is perfect for education and knowledge-based fields. It helps students, teachers, and professionals by providing fast, reliable answers, making learning more efficient and effective.

Keywords: Generative AI, RAG Chatbot, Embeddings, LangChain, Chroma, E-learning.

1. Introduction

Artificial Intelligence has most impactful innovation in recent years and it transforms how machines interact with human language. Some of the notable innovations are Generative AI models and Large Language Models. These technologies, such as chatbots, virtual assistants, and content-generation systems have been used to generate accurate responses to

DOI: https://doi.org/10.36548/jismac.2024.4.005
Received: 11.12.2024, received in revised form: 02.01.2025, accepted: 25.01.2025, published: 06.02.2025

user queries. One notable example is ChatGPT, a product of OpenAI. It is a Large Language Model trained on a vast amount of data and is used to generate accurate responses to user queries [2]. It is a general-purpose model that provides responses to various queries by retrieving information from its trained data, websites, articles, and other sources. Sometimes, the system may fetch irrelevant information, resulting in fabricated responses that are misleading and not useful to users [4-6]. To tackle this issue, Retrieval-Augmented-Generation (RAG) concepts offers a promising solution, it combines the retrieved content with the generative model to give the accurate responses which is an essential for delivering reliable information. LangChain is a framework used to streamline the integration of LLM models. It provides functions and methods for extracting text from PDFs, processing documents, and more. By using these technologies, the proposed system has been built for the education industry as an e-learning assistant to provide the most accurate responses to the students compared to the ChatGPT [7-9]. This RAG-based approach retrieves specific content from trusted educational materials (like PDFs).It focuses on delivering context-specific information for subjects like "Artificial Intelligence," unlike general AI systems that provide broad knowledge. The system is very adaptable, allowing educators to update the content, by providing faster, domain-specific responses. This makes the proposed system better suited for e-learning applications, ensuring students get the most accurate and up-to-date information [10-12].

2. Related Work

Recent research has focused on improving Retrieval-Augmented Generation (RAG) chatbots for specialized applications. Kulkarni et al. [1] used reinforcement learning to optimize RAG models, making them more accurate for domain-specific tasks like education and healthcare. Vidivelli et al. [14] combined LangChain with RAG to create efficient custom chatbots, highlighting their ability to handle large datasets effectively. Bora and Cuayáhuitl et al.[15] analyzed RAG-based chatbots in medical settings, showing their reliability for answering complex healthcare questions. Chaubey et al.[13] compared RAG with fine-tuning and prompt engineering, finding RAG better suited for diverse and dynamic datasets. Kang et al. [16] introduced Prompt-RAG, a vector-free approach for niche domains like Korean medicine, which reduces computational costs while maintaining accuracy. Quidwai and Lagana et al. [17] developed a RAG chatbot for precision medicine, demonstrating its potential

to provide precise, context-aware responses in healthcare. These studies show that RAG is good at giving accurate answers and focusing on specific topics. However, it also has challenges, as it requires high computing power, and high-quality data to work well.

3. Methodology

This section outlines the methodology used to develop the e-learning assistant RAG chatbot, detailing the processes involved in text extraction, embedding generation, data retrieval, and response generation.

3.1 PDF Extraction

The initial process involves extracting the text from the pdf document titled "Understanding Artificial Intelligence: The Future of Technology" which contains of 10 chapters that explores various aspects of Artificial Intelligence. The loader method used is LangChain's PyPDF loader which is a used to load the pdf document into memory and extract text content from the pdf document.

3.2 Document Splitting

After extracting the text, LangChain's RecursiveCharacterTextSplitter is used to break the large document into smaller manageable parts. This is important because working with smaller chunks of text makes it easier to handle. For example, setting a chunk_size of 1000 characters splits the document into sections of no more than 1000 characters each. This makes it simpler for the system to process and analyse the content effectively than the large document. The document splitting is shown in the Figure 1.

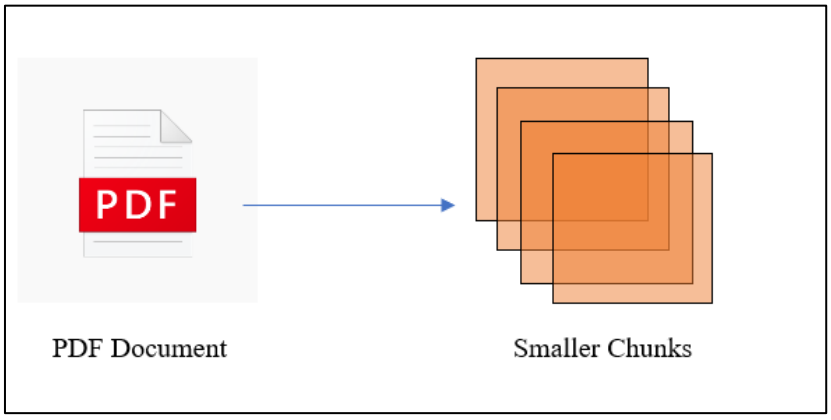

Figure 1. Document Splitting

3.3 Embedding Generation

The next step involves turning the extracted text into numerical vectors that is used to capture the meaning of the text. This process, is known as embedding generation. In this study, GoogleGenerativeAIEmbeddings is used to create a vector for each section of text. These vectors encapsulate the meaning of the entire section, considering all the words, sentences, and context within it. This approach helps the system grasp the deeper meaning of the text as a whole, rather than focusing on individual words alone. The embedding generation is shown in Figure 2.

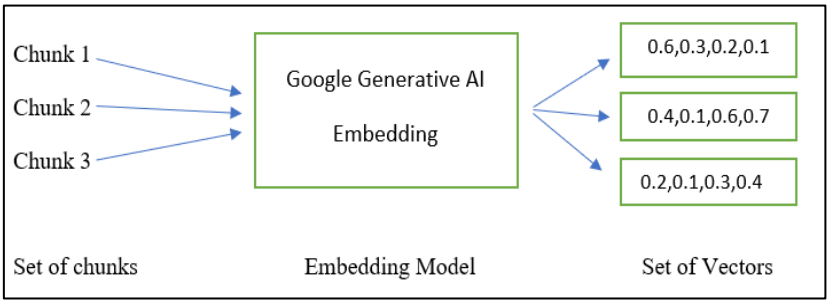

Figure 2. Embedding Generation

3.4 Vector Database

The next step involves storing the vectors that represent the meaning of the data in a database. A vector database is used to manage data in the form of vectors, which are numerical representations of various types of data such as text or images. For this study, the Chroma Vector Database is used to store the vectors generated during the embedding process. This database organizes the vectors and makes it easy to retrieve them when needed, allowing the system to quickly access the meaning behind the data whenever required. The vectors that are stored in the vector database is shown in Figure 3.

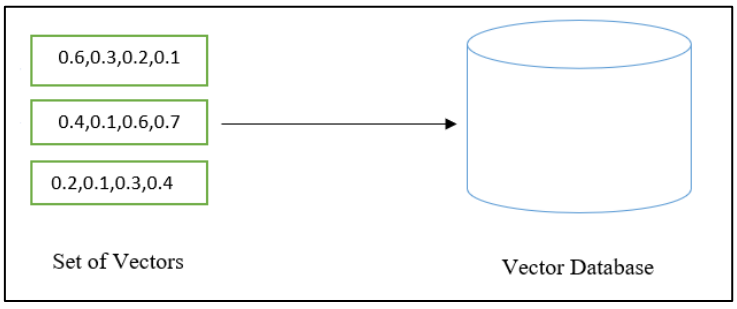

Figure 3. Vector Database

3.5 Query Processing

The process begins when a user submits a query that contains of text data. First, the query is transformed into a numerical format that is embedding using GoogleGenerativeAIEmbeddings, it captures the meaning behind the user's question. The embedding is then used to search the Chroma Vector Database, where it retrieves relevant chunks of information by comparing the query's vector with stored vectors that have similar semantic content. Once the relevant data is identified, Google's Gemini-1.5-Pro generative model is used to generate a response to the user interface. The model takes the both retrieved-context and the user's query to generate a precise and accurate answer. The Google Gemini is trained to give the response based on the following prompt:

"You are an assistant for question-answering tasks. Use the following pieces of retrieved context to answer the question. If you don't know the answer, say that you don't know. Keep the answer concise (maximum of 3 sentences)."

This methodology ensures that responses are accurate, relevant, and based on the user's query as well as the contextual data retrieved from the vector database. The workflow of this process is illustrated in Figure 4.

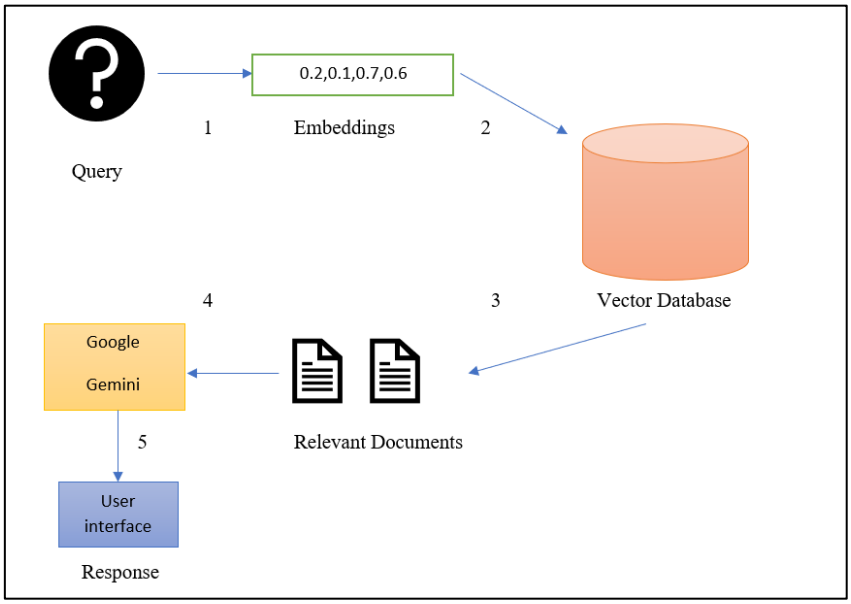

Figure 4. User Query Workflow

4. Results and Findings

This section presents the user interface and results from the RAG chatbot in action. The frontend of the user interface is created using React and bootstrap, a CSS framework has been used for the styling of the components such as padding, margin, border, background colour, position of the fields etc. The Figure 5 shows the RAG chatbot's front-end interface.

Figure 5. RAG Chatbot – E-learning Assistant

The interface includes a "Type your message here" field, allowing users to enter their queries. If a user enters a question, for eg: "Working of GAN brief?", the query is send to the backend for further processing. In the Figure 6, the typing indicator in the interface shows that the entered query is processing.

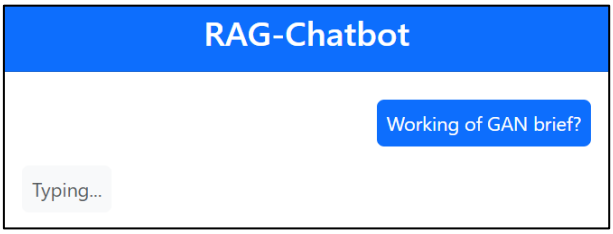

Figure 6. Chatbot Processing the Query

Flask, a Python framework used to establish the connection between the frontend and backend. The user query is in text format and it is converted into numerical format as vectors or embedding. Here the Google Generative AI Embedding act as the embedding machine to convert into numerical format. The embedding searches the Chroma database to find relevant

information. Then, Google's Gemini-1.5-Pro model uses this info to give a precise answer. The response is given in 3 lines because Google Gemini is trained to do it this way. This helps the model give clear and consistent answers.

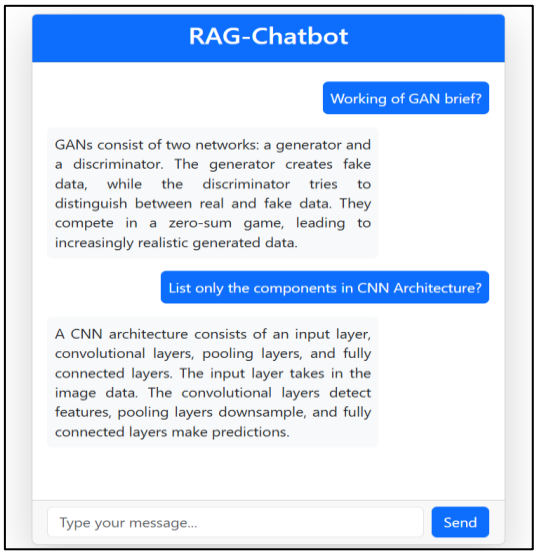

Figure 7. E-learning RAG Chatbot

Figure 7 displays the generated response in the RAG chatbot interface. Comparing the chatbot's response with the content from the PDF, it is evident that the generated answer aligns perfectly with the document's content. Figure 8 shows the corresponding content in the PDF for comparison.

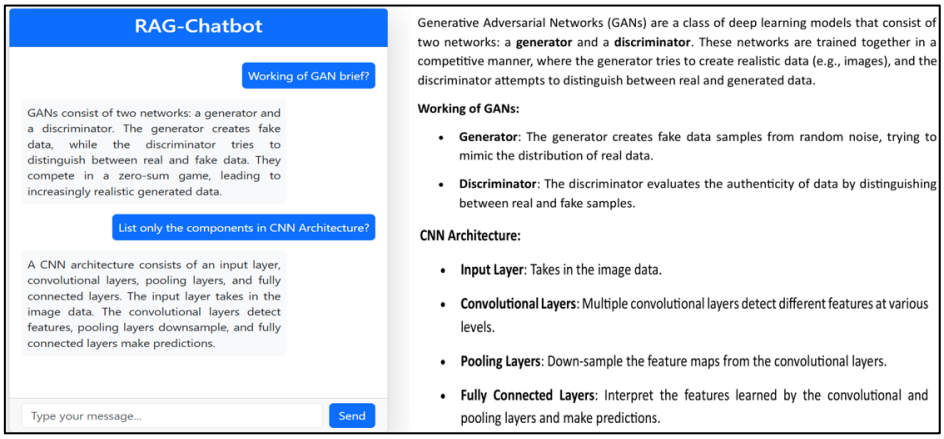

Figure 8. Comparison between Generated Response and Content in the PDF

While the wording may vary slightly, the key information remains consistent, demonstrating the RAG model's ability to deliver accurate and reliable answers.

5. Conclusion

This study looks at how artificial intelligence can improve automation in e-learning, especially in Artificial Intelligence. It focuses on turning PDF content and user questions into vectors, which are then processed by the RAG chatbot to give accurate answers. This method ensures the chatbot gives precise information, making it useful for e-learning. As AI gets better, these systems will become more personalized and efficient. This approach can also be used in other areas, like restaurant websites or sales sites, to answer customer questions. The system can be changed to fit different needs, making it flexible and easy to scale.

References

[1] Kulkarni, Mandar, Praveen Tangarajan, Kyung Kim, and Anusua Trivedi. "Reinforcement Learning for Optimizing RAG for Domain Chatbots." arXiv preprint arXiv:2401.06800 (2024).

[2] Jaiswal, Anuj, Garima Tiwari, Aakash Jha, Rushikesh Mangulkar, and Pushpi Rani. "Retrieval Augmented Generation Approach for Multipdf Chatbot using LangChain." In 2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA), IEEE, 2024. 1-6.

[3] Dhoni, Pan Singh, Saurabh Shukla, and Jagjot Bhardwaj. “Strategies for Integrating Generative AI in Industrial Settings.” International Journal of Computer Trends and fTechnology (IJCTT) 72, no. 7 (2024).

[4] Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 (2023).

[5] Rathod, Priyank Jayantilal. “Efficient Usage of RAG Systems in the World of LLMs.” Engineering 6, no. 4 (July-August 2024): Published July 10, 2024.

[6] “Retrieval-Augmented Generation Approach: Document Question Answering Using Large Language Model.” International Journal of Advanced Computer Science and Applications (IJACSA) 15, no. 3 (2024).

[7] Vincent, Christo, Allen Antony, Athul Asok, Abin Antony, and Anita Brigit Mathew. “Interactive VR Using RAG in Education.” International Journal of Research Publication and Reviews 5, no. 4 (April 2024): 9953–9956.

[8] Meduri, Karthik, Geeta Sandeep Nadella, Hari Gonaygunta, Mohan Harish Maturi, and Farheen Fatima. “Efficient RAG Framework for Large-Scale Knowledge Bases.” Journal Name 9, no. 4 (April 2024): h613–h622.

[9] Singh, J. "Combining Machine Learning and RAG Models for Enhanced Data Retrieval: Applications in Search Engines, Enterprise Data Systems, and Recommendations." J. Computational Intel. & Robotics 3, no. 1 (2023): 163-204.

[10] Akheel, Syed Arham. “Fine-Tuning Pre-Trained Language Models for Improved Retrieval in RAG Systems for Domain-Specific Use.” International Journal For Multidisciplinary vol 6, no. 5 (September-October 2024). 1-10.

[11] Muludi, Kurnia, Kaira Milani Fitria, Joko Triloka, and Sutedi. “Retrieval-Augmented Generation Approach: Document Question Answering Using Large Language Model.” International Journal of Advanced Computer Science and Applications (IJACSA) 15, no. 3 (2024).

[12] Singh, Jaswinder. “Understanding Retrieval-Augmented Generation (RAG) Models in AI: A Deep Dive into the Fusion of Neural Networks and External Databases for Enhanced AI Performance.” Journal of Artificial Intelligence Research 2, no. 2 (2022).258-275

[13] Chaubey, Harshit Kumar, Gaurav Tripathi, Rajnish Ranjan, and Srinivasa K. Gopalaiyengar. “Comparative Analysis of RAG, Fine-Tuning, and Prompt Engineering in Chatbot Development.” IEEE Xplore, September 30, 2024

[14] Vidivelli, S., Manikandan Ramachandran, and A. Dharunbalaji. “Efficiency-Driven Custom Chatbot Development: Unleashing LangChain, RAG, and Performance-Optimized LLM Fusion.” Computers, Materials & Continua 80, no. 2 (2024): 2423–2442.

[15] Bora, Arunabh, and Heriberto Cuayáhuitl. “Systematic Analysis of Retrieval-Augmented Generation-Based LLMs for Medical Chatbot Applications.” MDPI, October 18, 2024.

[16] Kang, Bongsu, Jundong Kim, Tae-Rim Yun, and Chang-Eop Kim. "Prompt-RAG: Pioneering Vector Embedding-Free Retrieval-Augmented Generation in Niche Domains, Exemplified by Korean Medicine." arXiv preprint arXiv:2401.11246 (2024).

[17] Quidwai, Mujahid Ali, and Alessandro Lagana. "A RAG Chatbot for Precision Medicine of Multiple Myeloma." medRxiv (2024): 2024-03.