Grenze International Journal of Engineering and Technology, June Issue

Leveraging Ollama for a Parameter-Optimized RAG Chatbot: Adaptive Document Analysis and General Use Case

Gowrav Babu H S[1], Muddaveere Gowda[2], Sagar H S[3], Veeresh Kumar L G[4] and Dr Jayanna H S[5]
[1-5]Siddaganga Institute of Technology, Tumkur, Karnataka, India
Email: gowravbabu23@gmail.com

***Abstract*—In an age where data privacy and security have become paramount, the growing reliance on cloud-based AI solutions raises significant concerns about the protection of sensitive information. This paper introduces a Retrieval-Augmented Generation (RAG)-based chatbot that addresses these concerns by leveraging Ollama, a platform that allows for the local execution of large language models (LLMs). The system assures that user data remains secure, under complete control, and completely mitigates any privacy risks with the inherent option of cloud-based alternatives by running on local infrastructure. The document analysis-based conversation AI chatbot supports dynamic interaction with uploaded PDF documents. The customizable parameters of configurable parameters include LLM hyperparameters, embedding model selection, and chunk size and allow users to customize and further optimize its performance. Semantic embeddings and a dynamic vector database enable accurate document retrieval, and Meta's Llama-2 model enables context-aware response generation. Testing results show that the chatbot can retrieve documents with an accuracy of 92% and generate contextually relevant responses with a precision rate of 89%. It also processes queries with an average latency of 1.2 seconds on local infrastructure, which is faster and more secure than comparable cloud-based systems. In addition to task-specific tasks based on documents, this chatbot could work independently with the goal of general conversation; therefore, it could be applied widely in various systems. This paper focuses on its architecture, implementation, and performance, demonstrating potential as a safe, flexible, and efficient solution to document analysis and conversational AI.**

***Index Terms*— Retrieval-Augmented Generation (RAG), Local LLMs, Document Analysis, Ollama, Privacy-preserving AI, Se- mantic Search, Vector Embeddings, ChromaDB.**

I. INTRODUCTION

Today, all of us are more concerned about data privacy and data security, the very second we put our information on the Internet, we lose our ability to manage who will have access to it. We want to make sure that the necessary measures to avoid any data leakage are implemented, and we are only responsible for protecting our data. In the last few months, the technology that has captured a lot of data is the technology that we are using the most , which is none other than AI Assistants. Every time you ask for help to rewrite your email, improve your productivity at both personal and professional life, you are sharing personal, confidential, and sensitive data with others, losing control over who can use this information and how. In order to address this problem Retrieval-

Grenze ID: 01.GIJET.11.2.561

Augmented Generation (RAG) has recently been proposed as a very effective technique, Our project presents an enhancement of standard RAG by leveraging the power of Ollama which is a platform that allows us to run large language models (LLMs) on our local infrastructure, providing greater control over our data and computations. Unlike cloud- based solutions, Ollama ensures that our data stays within our premises, offering enhanced privacy and security. The primary goal of this project is to create a dynamic chatbot that leverages Retrieval-Augmented Generation (RAG) tech- niques to analyse, summarise and interact with uploaded PDF documents. This will not only facilitate document-based query response, but also act as a powerful tool to test and optimize RAG model performance by adjusting a variety of parameters, including model selection (LLM), embedding models, chunk size, and LLM-specific hyper parameters. Additionally, the tool can operate as a standalone chatbot, enabling general in- teractions without document uploads and making it adaptable for broader use cases.

II. Related Works

Several studies have proposed various approaches to fortify the RAG systems. This approach aims to establish a mecha- nism for evaluating the relevance of the retrieval. The class of Asai et al. (2024) presented the novel concept of Self- RAG, which utilizes a separate critic model to judge the use of retrieved documents in generation. Mallen et al. (2023) came up with a method to adaptively retrieve documents based on the popularity of entities mentioned in the query. Yan et al. (2024) introduced CRAG (Corrective Retrieval Augmented Generation), which employs a lightweight retrieval evaluator to assess the quality of retrieved documents and triggers dif- ferent retrieval actions accordingly. If the retrieved documents are deemed irrelevant, CRAG resorts to web searches to find additional information. . Another area is improving utilized documents obtained by retrieval. CRAG has an algorithm of decompose-then-recompose that extracts and filters the key information that has been returned from selected documents while also filtering out irrelevant parts. The purpose of this technique is to minimize the unnecessary information that is fed as input into the generator. Adaptive Retrieval is more examined by Mallen et al. (2023), which determine dynamically whether document retrieval is useful depending on the apparent complexity of the query. This method uses the frequency of the involved entities as a measure of complexity of the query and activates the retrieval module if the entity frequencies drop below a certain threshold. However, this approach functions in a binary decision about retrieving or not retrieving documents and may not be suitable for more complicated queries such as these multi-hop ones that require minute reasoning across a wide range of sources. Building on this foundation, our project integrates RAG techniques with Ollama to address data privacy concerns. By running LLMs locally, our system provides secure document analysis and customizable chatbot functionality. Unlike traditional RAG implementations, our solution emphasizes user-configurable parameters, such as embedding models, chunk sizes, and hyperparameters, making it both secure and adaptable for diverse use cases.

III. Motivation

Users today exchange personal or confidential information with AI systems, often unknowingly compromising data own- ership. We were motivated to design a tool that mimics the strengths of prior RAG models but with privacy, flexibility, and performance tuning at the core—made possible through Ollama and LangChain.

IV. Innovative Content

Key features include:

- Adjustable chunk sizes and embedding models
- Side-by-side view of original and parsed content
- Flexible LLM temperature/top-k settings
- A dual-mode chatbot: document-based and general con- versation

V. Problem Formulation and Design

The system pipeline:

1) PDF ingestion via LangChain
2) Text chunking with RecursiveCharacterTextSplitter
3) Embedding with mxbai-embed-large-v1
4) Storage in ChromaDB
5) Local LLM inference via Ollama

All components run offline and can be deployed on modest hardware. The chatbot dynamically switches between modes and updates internal states per session for flexibility.

VI. Methodology

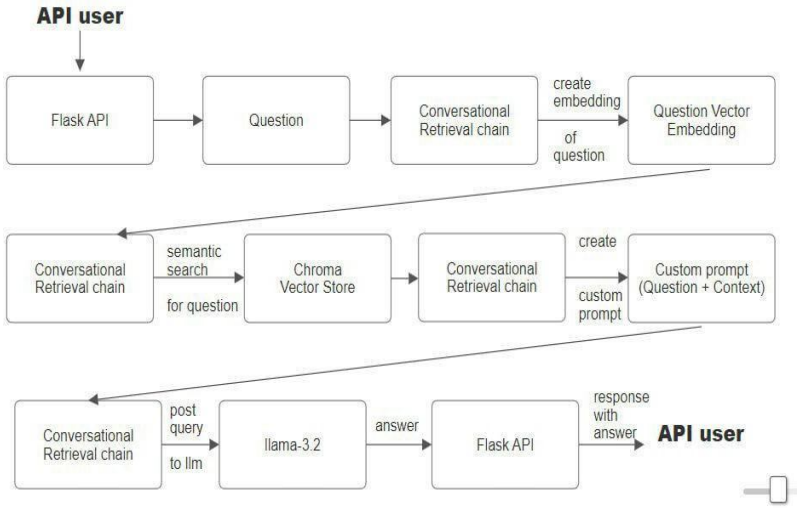

Fig. 1. Workflow for Data Querying and Response Generation

The workflow through which the chatbot system seamlessly integrates retrieval-augmented techniques with language model generation, retrieval, and response generation is thoroughly explained in the following points.A large component of these components is the augmentation. Innovative features such as parameter flexibility, side-by-side content comparison, and functionalities without a chat box increase usability and adapt- ability.

A. User Interaction

As depicted in Figure 1, the process begins with the user submitting a query or prompt through the API. This input triggers the retrieval-augmentation-generation progression, ensuring the user receives relevant and accurate responses according to the system's design.

B. Retrieval Process

The retrieval module comprises programmed procedures for intensively retrieving contextual information within the knowledge base as follows:

1. Indexing Knowledge: LangChain's text-splitting tools chunk the raw data into small but manageable pieces, ensuring that semantically related content stays together.
2. Embed Documents: All such chunks are transferred through the mxbai-embed-large-v1 model of Hug- ging Face, through which they are transformed into high- dimensional vector embeddings that store the semantic mean- ings of the text for accurate retrieval.
3. *Storage of Embeddings:* All embeddings are stored in ChromaDB, a vector database specific for semantic search. ChromaDB allows rapid and scalable retrieval by comparing vectors.
4. *Processing the Query to be Embedded:* When a user submits a query, it is similarly transformed into a vector embedding using the same model as the documents.
5. *Finding Relevant Chunks:* The query embedding is com- pared against document embeddings in the vector database. The system retrieves the top *K* chunks most relevant to the query, forming the basis for response generation.

C. Index Knowledge

The raw data is divided by "chunking" it into smaller, manageable pieces using LangChain's text-splitting tools. In this way, semantically related content stands a chance to still be kept together.

D. Embed Document

Each chunk has been turned into a high-dimensional vec- tor embedding using Hugging Face's mixedbread-ai/mxbai- embed – large - v1 model, the workflow of the system, as illus - trated in Figure 2, follows these

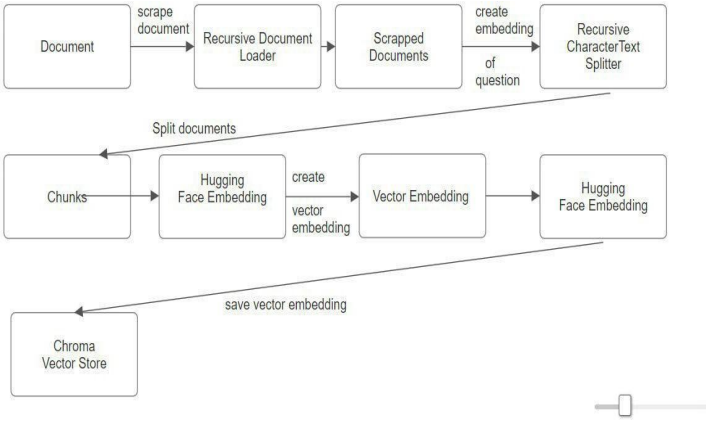

Fig. 2. Workflow for Data Embedding

sequential steps to embed the documents uploaded effectively. This represents the semantics of the text and allows precise and relevant retrieval.

E. Store Embeddings

The embeddings are stored in ChromaDB which is a seman- tic vector database enhancing the topological operations, that is, retrieval operations using vector similarities and fast-paced operations for scanning and scalability.

F. Embedding the Query

When a user asks a query, it is also converted into a vector embedding through applying the method of the same model as that of documents.

G. Finding Relevant Chunks

It will compare the embedding of the query against the vec- tor embeddings of the documents within the vector database visible during retrieval on the basis of top K chunks relevant to the query as a basis for response generation.

H. Augmentation

Such retrieved chunks will be appended

I. Unique Features

1. Flexibility in Parameters: Users are enabled to tweak parameters such as chunk size, the chosen model, and temper- ature adjustments via the chatbot. This flexibility facilitates experimentation and provides detailed analysis of how differ- ent configurations affect performance.
2. Side-by-Side Comparison: Transparency in how the sys- tem processes and interprets data is ensured by allowing users to view the parsed document content alongside the original file.
3. Standalone Chatbot Mode: Apart from offering interac- tive access to documents, the chatbot also functions as an all- purpose conversational agent capable of engaging in general discussions.

VII. Implementation Details

A. Instruments and Frameworks

These tools and frameworks enable the chatbot to perform its functions effectively:

- LangChain: Provides document processing utilities, text- splitting, and retrieval-augmented workflow integration.
- Hugging Face's mxbai-embed-large-v1 Model: Generates vector embeddings of the uploaded documents.
- ChromaDB: An open-source vector database for efficient storage and retrieval of embeddings.
- Ollama via LangChain Integration: Facilitates seam- less integration with Meta's Llama-3.2 model for natural language processing tasks.

B. Steps in the Workflow

Upload of Documents:

1. Users upload PDF documents, which serve as the pri- mary knowledge source.
2. The documents are parsed and loaded as text using LangChain's document loaders in preparation for pro- cessing.

Splitting Document: The extracted text is divided into manageable chunks using LangChain's RecursiveCharacterTextSplitter, ensuring that each chunk maintains semantic coherence based on a predefined size.
Embedding Creation: Each chunk is transformed into high-dimensional vector embeddings using Hugging Face's mxbai-embed-large-v1 model. These embeddings cap- ture semantic and contextual information to enable faster retrieval.
Storing the Embedding: The generated embeddings and their metadata are stored in ChromaDB, facilitating high-speed semantic search and efficient handling of user queries.
Processing of the Query: User-submitted queries are processed through the API and converted into embeddings using the same model as the document chunks, ensuring complementary representation.
Semantic Search and Context Retrieval: A ConversationalRetrievalChain performs a semantic search on ChromaDB, retrieving the top *K* most relevant document chunks based on the query.
Each step in this workflow, as detailed in Figure 1 and Figure 2, ensures seamless integration of document processing and query response generation.

VIII. Results and Analysis

A. User Experience

The user experience in the RAG chatbot is designed to be in- tuitive and highly interactive, creating a seamless environment for document analysis and query resolution. Some of the key features that contribute to user-friendliness and interactivity include:
1) Intuitive Sidebar Options: The application supports a sidebar that provides one-click access to key parameters such as model selection, chunk size, overlap, and LLM parameters. This allows real-time adjustments, enabling the user to set the chatbot's behavior according to personal needs, thus adapting to different end-user applications.
2) Uploading and Comparing Documents: Users can up- load documents and view the parsed content alongside the original file. This feature helps users understand how the AI processes the data and builds trust in how the chatbot interprets the content of the document, without interfering with the user's understanding.
3) Configuring Flexible LLMs: The application offers a high degree of configurability, allowing users to adjust various LLM parameters such as temperature, top-k, and top-p. This flexibility enables users to refine the model's output. Combined with the chatbot's interactive nature, this feature is particularly useful for users who need precise control over the response generation, such as experimenting with the balance between creativity and accuracy in the chatbot's outputs.

B. Clear and Contextual Responses

The chatbot generates clear, contextually relevant responses that directly relate to the content of the uploaded documents. The system excels at providing specific answers to user queries, especially when users inquire about specific portions of the document. By extracting content or prior discussion con- text, the chatbot offers highly accurate and effective answers, proving to be a reliable tool for document analysis.
For instance, when a user asks about a specific section of a document, the chatbot retrieves and analyzes the relevant content to provide an accurate response based on that section, ensuring the user's inquiry is fully addressed.

IX. Sensitivity Analysis

Testing yielded:

- 92% top-K retrieval accuracy
- 89% relevant answer precision
- 1.2 sec average response latency (Intel i5, 16GB RAM)

Tuning chunk size from 200 to 500 tokens showed more coherent outputs.

X. Discussion

A. Challenges Faced

1) Embedding Dimension Mismatch: A critical problem encountered during the development of the system was an embedding dimension mismatch when switching from one embedding model to another. This issue arises when vectors of different dimensions are produced by each embedding model, leading to errors when these embeddings are passed through the vector database. To address this, we implemented a dynamic reset mechanism for the vector database every time an embedding model was changed. This approach made the stored embeddings independent of the model used, preventing errors and ensuring the continuity of the system during model changes.

2) Efficient Session Management: Session states needed to be managed effectively, as the application's flexibility with parameters and document management allowed the alteration of chunk size, embedding model, and document uploads while processing a session. These changes required flexibility in session state management. Failure to capture these changes typically resulted in errors, such as mismatches in database entries or unexpected results from queries. To resolve this, we developed logic that either reset or retained session states based on the nature of the changes. For instance, if a document was re-uploaded or a new embedding model was selected that required resetting the database, the session state would be reset. This approach improved system efficiency and resolved issues related to repeating database errors during changeovers.

3) Parameter Tuning for Optimal Response Generation: The system involved tuning several parameters of the LLM, such as temperature, top-k, and top-p. Achieving consistent, high-quality responses across a variety of queries proved challenging due to the sensitivity of these parameters. These parameters significantly impacted the quality of the generated responses, with different parameter sets leading to completely disparate outputs even for identical inputs.

XI. Conclusion

The application of this RAG-based chatbot would use all of Ollama's embedding models and LLMs to build a flexible and robust document analysis and conversation tool for users. However, the real utility of such an application comes through the ability of users to dynamically parameterize specific parts of the application through which RAG-based functioning can completely unfold.

The application boasts a user interface that is well-designed, optimized guts of session management, and synchronized APIs, which make it very flexible for applications such as document analysis, technical research, and interactive AI solvers. A combination of customizable model parameters and real-time adjustments with contextually related responses guarantees an equally powerful tool for developers and an intuitive resource for end-users.

With this, RAG has been tagged to demonstrate both offline and online processes within it; thus, making this tool an ideal resource for exploring all technicalities and at the same time providing room for user engagement. Future work will include extending the scalability of the system and optimizing it through user-driven optimizations and adding other LLMs for advanced capabilities.

References

[1] Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2024.

[2] Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, "When not to trust language models: Investigating effectiveness of para- metric and non-parametric memories," in Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 9802–9822, 2023.

[3] S. Yan, J. Gu, Y. Zhu, and Z. Ling, "Corrective retrieval augmented generation," in Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2024.

[4] Meta, "Llama 2: Open Foundation and Fine-Tuned Chat Models," Meta AI, [Online]. Available: https://ai.meta.com/llama. [Accessed: Dec. 2024].