


DocBot

Enhancing Document Retrieval and Generation with RAG

By: Charmi Chokshi


Agenda


- Problem Statement
- Solution Overview
- System Overview
 - RAG Framework
 - Technologies Used
 - DocBot Flow
- Features of DocBot 
- Methodology
- Demo
- Future Work
- Conclusion
- Q&A


Problem Statement

- Large Language Models (LLMs) suffer from issues such as **Knowledge cut-offs** and **hallucinations** and they are not exposed to private data.
- Traditional search engines and FAQ systems fail to provide contextually relevant and natural language responses.


 **You**
who is the current CEO of Twitter?

 **ChatGPT**
As of my last update in January 2022, the CEO of Twitter was Parag Agrawal. However, executive positions can change over time, so I would recommend verifying this information with up-to-date sources to confirm the current CEO of Twitter.

 **You**
how many M's are in 'Weather'?

 **ChatGPT**
In the word "Weather," there is 1 letter "M."

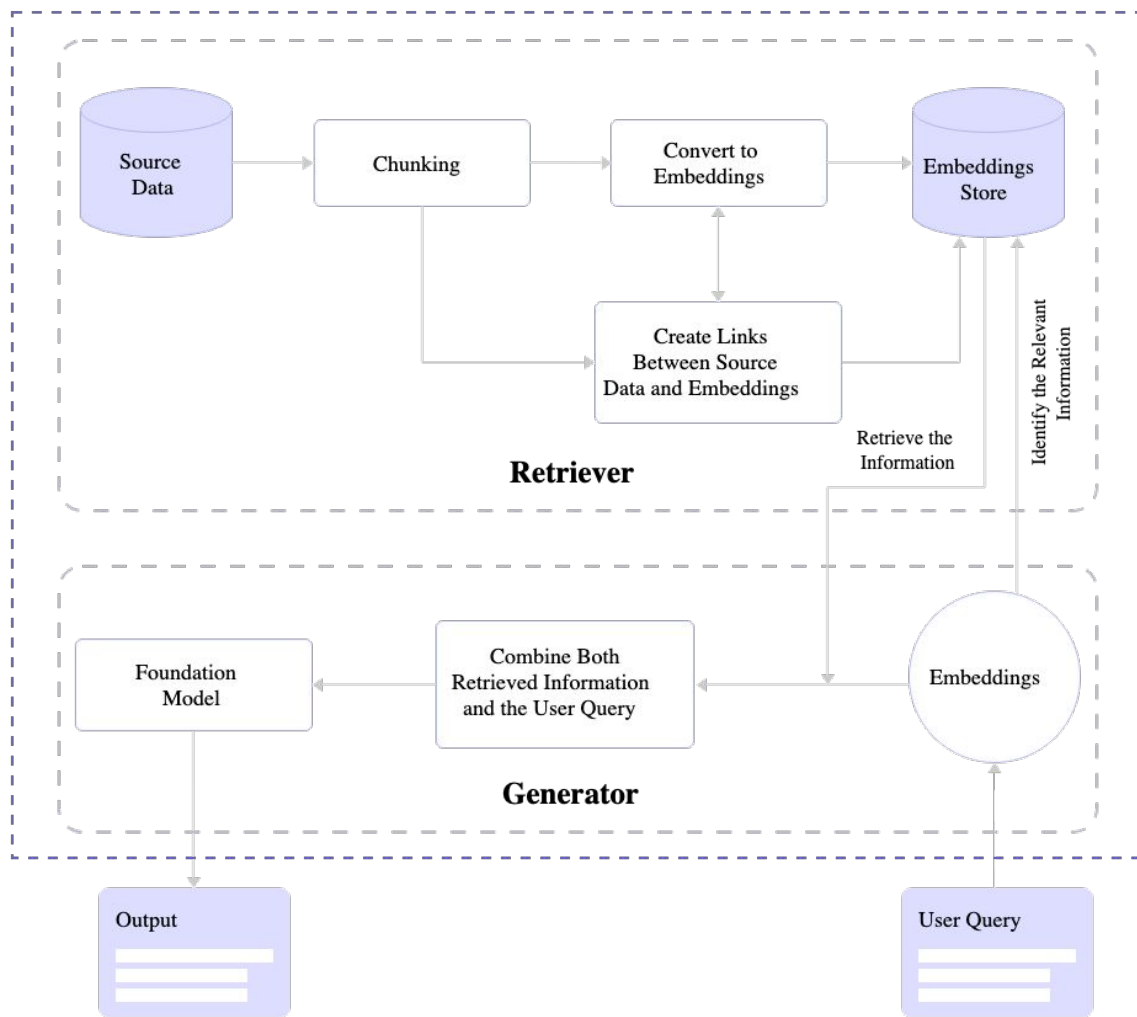
Solution Overview

- There is a need for an intelligent system that can provide **accurate and relevant answers based on specific document context while leveraging the language understanding capabilities of LLMs.**
- DocBot  is a Retrieval-Augmented Generation (RAG) chatbot that combines the power of LLMs with a searchable vector store created from user-provided documents.
- This approach ensures high-quality, contextually relevant responses without hallucinations, bridging the gap between traditional search engines and language models.

System Overview

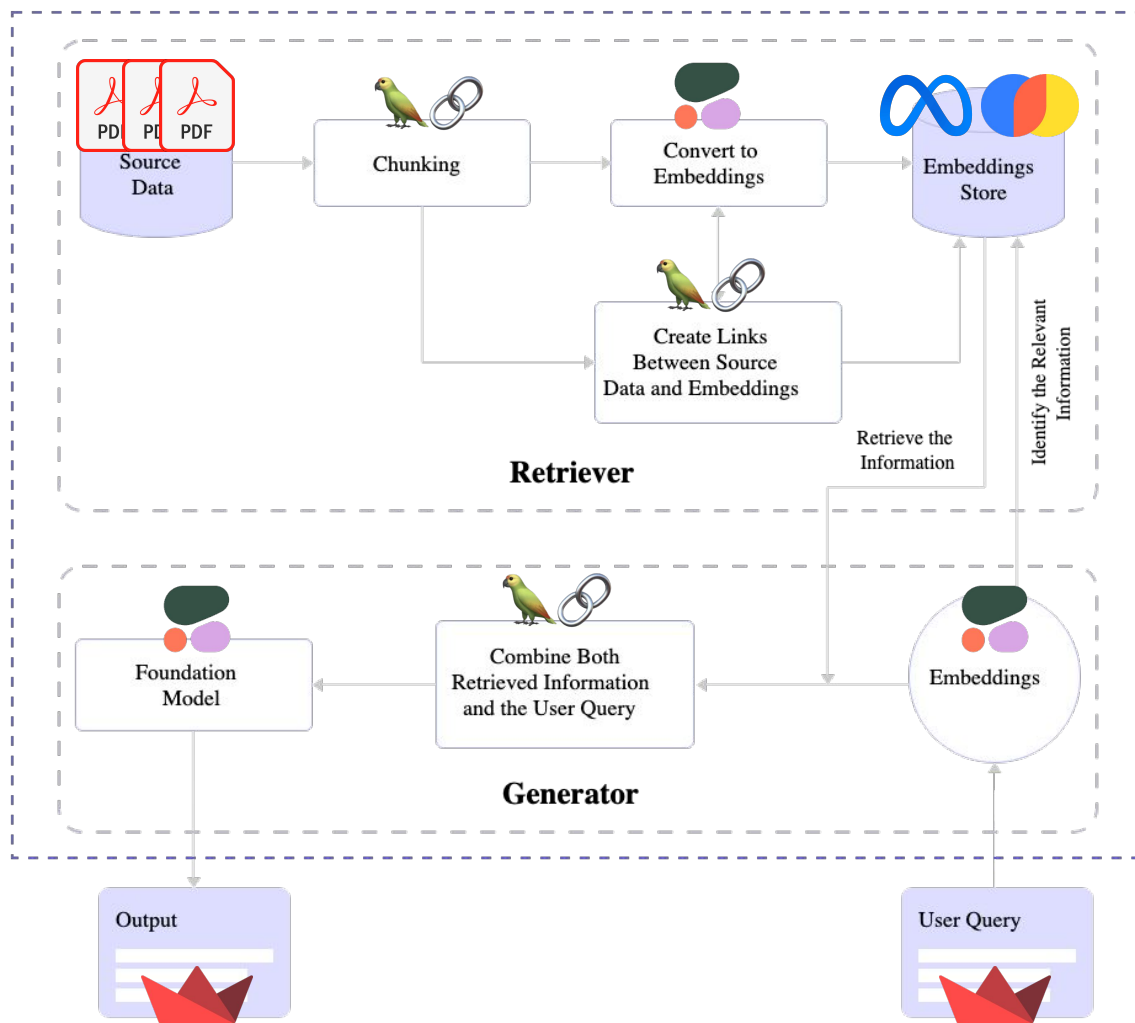
RAG Framework

- **Retriever:** Responsible for retrieving relevant information from a large knowledge base.
- **Generator:** Responsible for taking the retrieved information and generating coherent and contextually relevant responses or text.



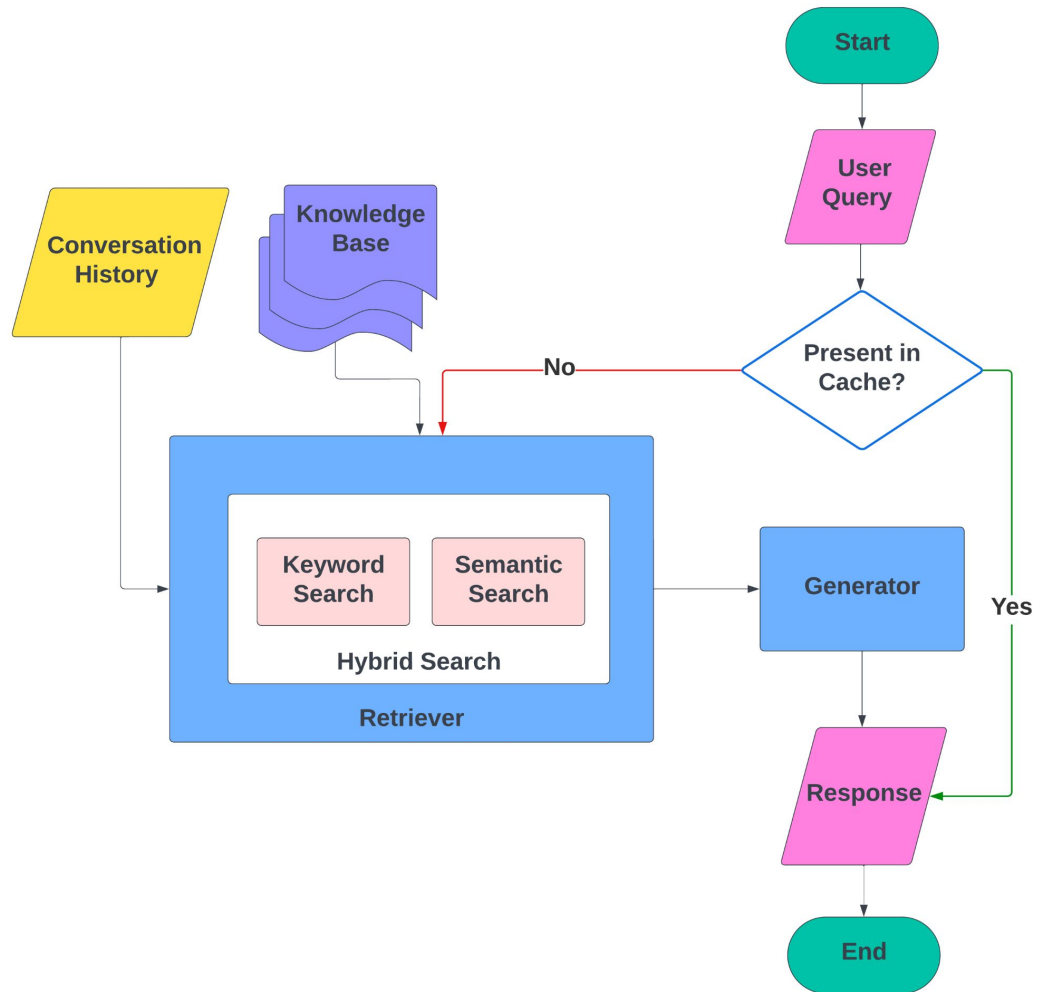
Technologies Used

- **LLM:** ChatCohere
- **Embeddings:** Cohere embed-english-v3.0
- **Doc Processing:** Langchain
- **Vector Store:** FAISS, Chroma
- **Web App:** Streamlit
- **PDF Processing:** PyPDF2



DocBot Flow

- **Caching Embeddings:** Embeddings are cached to avoid recomputing.
- **Hybrid Vector Search:** Combines keyword search (BM25 Algorithm) and semantic search (FAISS) for efficient retrieval.
- **In-Memory Caching:** Caches responses for previously asked queries to improve performance.
- **Conversational System:** Remembers and considers the context of previous questions and answers.



Features of DocBot

- Accurate and contextually relevant answers based on user-provided documents.
- No hallucinations or factual inconsistencies due to the use of RAG.
- Efficient and scalable document processing and retrieval.
- Conversational capabilities with context tracking.
- Flexible and extensible architecture.

Features of DocBot

×


Uploader:

Upload PDF Files, Configure options and Click on the Submit & Process Button

Drag and drop files here

Limit 200MB per file

Browse files

 Charmi Chokshi Resum...
143.6KB

×

Select Storage Type

☐ Chroma

☒ FAISS

Select Advanced Features

☐ Cache Embeddings

☐ Conversational System

☒ Hybrid Search + Cache

☐ None

Submit & Process

Deploy

DocBot : Chat with PDFs and get instant insights!

About


^

This chatbot is built using the Retrieval-Augmented Generation (RAG) framework, leveraging Cohere Model and Embeddings. It processes uploaded PDF documents by breaking them down into manageable chunks, creates a searchable vector store, and generates accurate answers to user queries. This advanced approach ensures high-quality, contextually relevant responses for an efficient and effective user experience. (It will not hallucinate an answer if its missing context.)


How It Works


^

- Upload Your Documents:** You can upload multiple PDF files at once. Press Submit & Process.
- Ask a Question:** After processing the documents, ask any question related to the content of your uploaded documents for a precise answer.

Start Chatting 

Who is Charmi?

Ask 

DocBot : Charmi Chokshi is the name of the person whose resume is provided in the context. Charmi Chokshi is a skilled professional with a demonstrated history of working in the research and development industry. Charmi has a Master of Science (MSc) in Computer Science and Machine Learning from the University of Montreal & Mila. According to her resume, she has extensive experience as a Machine Learning Engineer and Research Engineer Intern at Amazon Web Services.

Execution time: 1.84 seconds

Methodology

Methodology: Document Processing

- Extract the text content and metadata from the PDF.
- Split them into **manageable chunks** for efficient processing and retrieval.
- The "RecursiveCharacterTextSplitter" class from LangChain is used.
 - "**chunk_size**" determines the maximum length (in characters) of each text chunk.
 - "**chunk_overlap**" specifies the number of overlapping characters between consecutive chunks.
- The resulting chunks are stored as Document objects in LangChain, ready for the next step.

chunk_overlap=x% | Total Chunks=9

Laser Inertial Fusion Energy

LIFE, short for Laser Inertial Fusion Energy, was a fusion energy effort run at Lawrence Livermore National Laboratory between 2008 and 2013.

LIFE aimed to develop the technologies necessary to convert the laser-driven inertial confinement fusion concept being developed in the National Ignition Facility (NIF) into a practical commercial power plant, a concept known generally as inertial fusion energy (IFE).

LIFE used the same basic concepts as NIF, but aimed to lower costs using mass-produced fuel elements, simplified maintenance, and diode lasers with higher electrical efficiency.

Background

Lawrence Livermore National Laboratory (LLNL) has been a leader in laser-driven inertial confinement fusion (ICF) since the initial concept was developed by LLNL employee John Nuckols in the late 1950s. The basic idea was to use a driver to compress a small pellet known as the target that contains the fusion fuel, a mix of deuterium (D) and tritium (T).

If the compression reaches high enough values, fusion reactions begin to take place, releasing alpha particles and neutrons. The alphas may impact atoms in the surrounding fuel, heating them to the point where they undergo fusion as well. If the rate of alpha heating is higher than heat losses to the environment, the result is a self-sustaining chain reaction known as ignition.

Methodology: Vector Store

- Flexible Storage: Supports both FAISS and Chroma vector stores.
- Cohere's "embed-english-v3.0" model used for generating embeddings
- **Embeddings** computed for text chunks and user queries
- **FAISS index** or **Chroma vector store** created
- Text chunk embeddings stored for efficient similarity search



Methodology: Cache

- "CacheBackedEmbeddings" used for **caching embeddings**
- Avoids recomputing embeddings for previously processed text
- Improves performance and efficiency
- **In-Memory Caching**
 - Utilized for caching embeddings in memory
 - Provides fast access to cached embeddings
 - Suitable for shorter-term caching during runtime
 - Useful when the **cache size is relatively small**
- **Persistent Caching**
 - LocalFileStore explored for persistent caching on disk
 - Embeddings stored in './cache/' directory
 - Allows caching across multiple runs and sessions
 - Usefulness in scenarios where the **cache size is large**

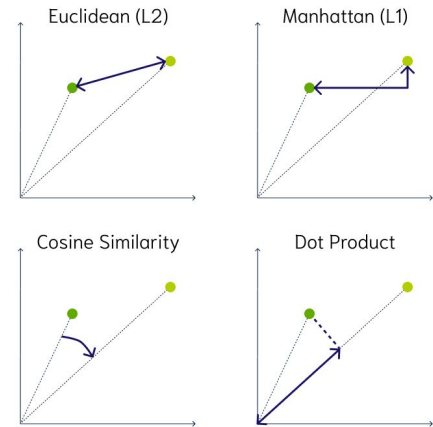
Methodology: Searching Techniques

- **Similarity Search**

- FAISS: "similarity_search" finds nearest neighbors in the vector space based on L2 (Euclidean) distances or dot products
- Chroma: similarity_search retrieves documents with embeddings closest to the query embedding

- **Hybrid Retrieval Approach**

- "EnsembleRetriever" used for combining vector and keyword search
- FAISS for semantic vector search
- BM25 Algorithm for keyword-based search
 - Ranking function used for keyword-based search, which ranks documents based on TF-IDF



$$\text{score}(D, Q) = \sum_{t \in Q} \frac{f_{t,D} \cdot (k_1 + 1)}{f_{t,D} + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})} \cdot \log \frac{N - n_t + 0.5}{n_t + 0.5}$$

Annotations for the formula:

- sum the scores for each query term (points to the summation symbol \sum)
- term frequency saturation trick (points to $f_{t,D}$)
- forget about this: it doesn't affect score relationships so Lucene took it out (points to $k_1 + 1$)
- adjust saturation curve based on document length (points to $\frac{|D|}{\text{avgdl}}$)
- probabilistic flavor of IDF: Lucene adds a 1 inside the log, making it basically the same as traditional IDF (points to $n_t + 0.5$)

Methodology: Conversational History

- Enables the system to have a coherent, contextual conversation
- Conversation Buffer Memory
 - "ConversationBufferMemory" from LangChain used to store conversation history
 - Remembers previous questions and responses for context
- Conversational Retrieval Chain
 - "ConversationalRetrievalChain" combines retrieval and generation
 - Utilizes "load_qa_with_sources_chain" for answer generation
 - Leverages conversation history from "ConversationBufferMemory"

What is charmi's email id?

Ask ➤

DocBot 🧙: The context gives Charmi Chokshi's email ID as charmichokshi@gmail.com.

Execution time: 1.08 seconds

What is her github user name?

Ask ➤

DocBot 🧙: Charmi Chokshi's GitHub username is charmichokshi. It can be found in the resume's skills section, which lists various programming languages and frameworks alongside it.

Demo

Future Work

- Fine-tuning Google Gemma-2B on domain-specific documents using PEFT techniques like QLoRA for improved performance in specific domains (e.g., finance).
- Data curation and continuous learning to enhance the knowledge base.
- Exploration of other state-of-the-art LLMs and vector databases.
- PoC has been done:

<https://github.com/charmichokshi/Experiments-with-Gemma>

Conclusion

- Developed and Deployed a Chatbot using RAG framework.
- Explored various Caching and Vector Matching techniques for optimised retrieval of data from PDFs.
- These advanced approaches ensures high-quality, contextually relevant responses for an efficient and effective user experience.

References

- Code: <https://github.com/charmichokshi/DocBot/>
- App: <https://docbot-by-charmi.streamlit.app/>

Thank You!

Questions?