GOA UNIVERSITY

Course Code: CSI-509

Course Name: Deep Learning Lab

Project Report

RAG-based Legal Document Chatbot

Submitted by:

Avinesh Pambally (24P0620003)

Chaturthi Naik (24P0620004)

Class: MSc. Artificial Intelligence – Part 1

Academic Year: 2024–2025

1. Introduction

Legal documents can often feel overwhelming and hard to understand because they tend to be filled with complicated jargon, long sentences, and confusing formatting. That's why we're working on creating a chatbot that can help answer questions about legal papers. This tool uses a smart approach called Retrieval-Augmented Generation (RAG), which combines two powerful technologies. First, it uses semantic search to understand what you're asking and find the most relevant parts of the document. Then, it uses advanced language models to explain those sections in simple, everyday language. For example, if you ask, “What are my rights if my landlord refuses to return my security deposit?” The chatbot will look through the document to find specific information about deposits, then provide a clear summary that includes details like deadlines, possible penalties, and what you can do next. Unlike basic chatbots, this system stays focused on the specific document you’re asking about, which helps reduce mistakes and makes complicated legal terms easier to grasp. Overall, it’s designed to make legal documents more accessible, helping people understand contracts, policies, or rules with confidence and less stress.

2. Objective

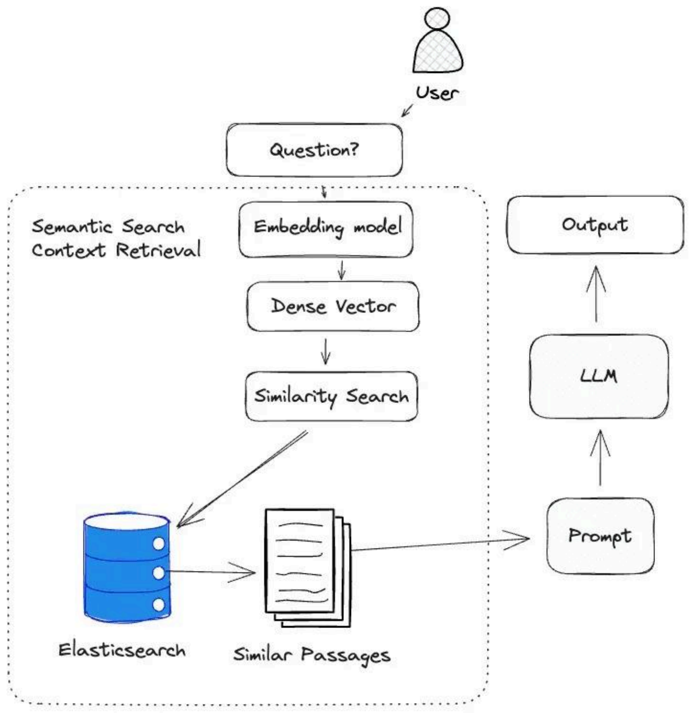
To develop an intelligent chatbot that:

- Accepts legal documents in PDF format as input,
- Converts them into semantically meaningful chunks and stores them in a vector database,
- Retrieves relevant chunks based on user questions using multi-query augmentation,
- Generates responses using a local LLM (gemma3:4b) based solely on the document context,
- Presents an interactive and user-friendly chat interface.

3. Tools and Technologies

Component	**Technology**
Orchestration	LangChain
Embedding & LLM	Ollama (gemma3:4b)
Vector Store	Chroma
Text Extraction	UnstructuredPDFLoader
Chunking	RecursiveCharacterTextSpillter
UI Interface	Gradio
Language	Python

4. System Architecture



4.1. Workflow

PDF Loading

A legal document is loaded using UnstructuredPDFLoader, which extracts text from unstructured PDF layouts.

Text Chunking
The extracted text is split into manageable pieces using RecursiveCharacterTextSplitter.

Embeddings
Chunks are embedded using OllamaEmbeddings with the nomic-embed-text model to create vector representations.

Vector Store
The vectors are stored in a local Chroma vector database.

Multi-Query Retrieval
A multi-query retriever uses the gemma3:4b model to reformulate the user query into multiple perspectives for better recall during search.

RAG Chain
Retrieved chunks are passed through a prompt template and processed by the gemma3:4b model using LangChain's Runnable interfaces.

Chat Interface
The chatbot is deployed through a Gradio-based UI that allows users to ask questions and view answers

4.2. Component Description

1. PDF Loading
The UnstructuredPDFLoader is a handy tool in the LangChain toolkit that helps you quickly pull out text and metadata from legal PDFs, even if they have tricky layouts or are just scanned images. It can handle both regular and encrypted PDFs and works with many formats, making it useful in lots of legal and professional situations. Thanks to built-in OCR features, it can process scanned documents so that both machine-readable and image-based PDFs turn into clean, organized text. It also keeps important details like the author, title, and creation date, which help when you're organizing and searching through legal files. The extracted information is returned in a neat, structured way, ready

for further processing, analysis, or integration with other LangChain tools. Overall, the UnstructuredPDFLoader is a key starting point whenever you need reliable, thorough data extraction from complex PDF files.

2. Text Chunking
The RecursiveCharacterTextSplitter is a tool that breaks large texts into chunks of about 3,000 characters each, with a 200-character overlap between sections. This overlap is really helpful because it keeps important legal details from getting lost when dividing the text. By overlapping, it makes sure that the flow of legal arguments, clauses, or references continues smoothly across different parts. This way, the system can understand and keep the connection between sections, making sure that even longer legal documents with cross-references stay clear and meaningful. Overall, this approach helps improve the accuracy of retrieving information, by preventing important context from being cut off or broken up too much.

3. Embeddings
OllamaEmbeddings turns chunks of text into numerical vectors using the 'nomic-embed-text' model, which is designed specifically for creating meaningful semantic embeddings that work well for search and retrieval tasks. These vectors are able to capture the underlying legal meanings and relationships, helping the system understand complex legal language and ideas. By converting each chunk into a vector, the model makes sure that even subtle legal terms and contexts are kept intact, so that it can accurately match user queries with the right parts of documents. This process is really important for enabling smart search features in retrieval-augmented generation (RAG) systems, as it allows the chatbot to compare the actual meaning of different text passages rather than just looking for matching keywords.

4. Vector Store
Chroma organizes vectors into searchable indexes with helpful metadata tracking. This means each piece of text can be linked to key details like

where it came from, its section, or other attributes of the document. Because of this setup, you can quickly find similar content compared to what you’re asking, using smart search techniques like HNSW for vector searches and filtering with SQL based on metadata. So, Chroma isn’t just good at understanding the meaning of your queries and matching them to relevant parts of documents, but it also lets you filter and sort results based on metadata. This way, you always get results that are not only accurate but also relevant to the specific context you're interested in.

5. Multi-Query Retrieval
The retriever creates 2 to 3 different versions of a query using gemma3:4b to help it overcome the limitations of matching keywords exactly. This way, it can catch a wider range of meanings and ideas, making the search more thorough. By rewriting the same question in different ways—whether with different phrases or angles—the system can find more relevant legal information. This includes clauses, laws, or case references that might not be caught with a simple keyword match. It’s like looking at the same problem from different perspectives, which helps reveal subtle connections and alternative ways to interpret the texts. Overall, this makes sure important details aren’t missed just because the keywords don’t match perfectly, helping the search adapt to different legal terms and what the user is really looking for.

6. RAG Chain
LangChain's pipeline merges retrieved chunks with the user's original question through a structured prompt template, ensuring that the context of the legal document is preserved and effectively utilized in the response generation process. This approach allows for seamless integration of multiple data sources and precise formatting, which is crucial for handling complex legal documents and queries. The gemma3:4b model then synthesizes coherent, document-grounded answers by leveraging the contextual information from the retrieved sections, providing responses that are both accurate and relevant to the specific legal question at hand. This workflow not only automates the extraction and summarization of legal information but also enhances the

reliability and transparency of the answers, making it easier for users to understand and trust the system's outputs.

7. Chat Interface
Gradio provides a web-based chat UI with conversation history and example questions, allowing users to interact with the chatbot in a familiar messaging format. The interface is intuitive and easy to use, making it accessible even for those without technical backgrounds. Users can view previous exchanges, refer back to earlier responses, and test the system with predefined example queries, all within the same session. Gradio also supports markdown formatting, enabling clear display of legal references, code, or emphasized text in responses. Additionally, the chat interface can be customized in appearance and functionality, and it can be shared via a public link or embedded as a widget on a website, making it convenient for both private and collaborative use.

5. Example Queries

- "What is the court's stance on irrevocable licenses in this case?"
- "What reasoning did the court provide regarding license revocation?"
- "Explain the key legal issues discussed in the judgment."

The system was able to return grounded and relevant responses based on the provided legal document, showcasing the accuracy of the RAG pipeline.

6. Results and Observations

- The chatbot accurately retrieves and summarizes legal content when asked case-specific questions.
- The use of multi-query reformulation improves the diversity and relevance of retrieved content.
- Running the gemma3:4b model locally through Ollama offers privacy and reduces dependency on cloud APIs.

- The Gradio UI provides an accessible way for users to interact with the chatbot.
- All answers are grounded strictly in the input document, ensuring trustworthy results.

7. Conclusion

This project effectively demonstrates how a Retrieval-Augmented Generation (RAG) approach can transform legal document analysis and question-answering systems. By strategically combining local LLMs like gemma3:4b, vector search through Chroma's efficient indexing, and workflow orchestration via LangChain's modular pipelines, the system delivers precise, contextually accurate responses that remain strictly grounded in complex legal texts. The architecture ensures answers are derived directly from document content, minimizing hallucinations while enabling natural language interactions with dense legal materials.

8. Future Enhancements

- Support for batch processing of multiple legal documents.
- Option for switching between multiple LLMs in real-time.
- UI enhancements such as document summarization and highlight extraction.
- Integration of regional languages for multilingual legal support.
- Logging and feedback collection to improve model performance.

9. References

- LangChain Documentation - https://docs.langchain.com/
- Ollama - https://ollama.com/
- Chroma Vector DB - https://www.trychroma.com/
- Gradio - https://www.gradio.app/
- Case Law: Ram Sarup Gupta vs Bishun Narain Inter College, 1987