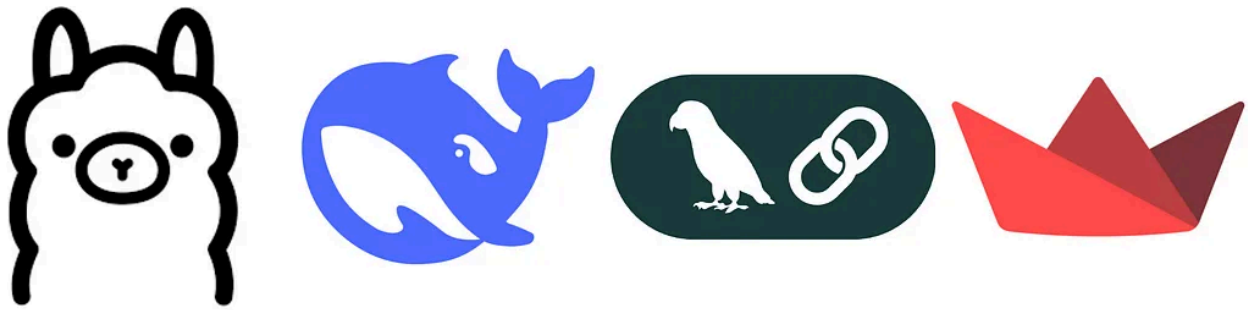
How to Build a Local RAG Using DeepSeek-R1, LangChain, and Ollama

Pedro Aquino · Follow
Published in GoPenAI
4 min read · Jan 29, 2025

Listen Share More

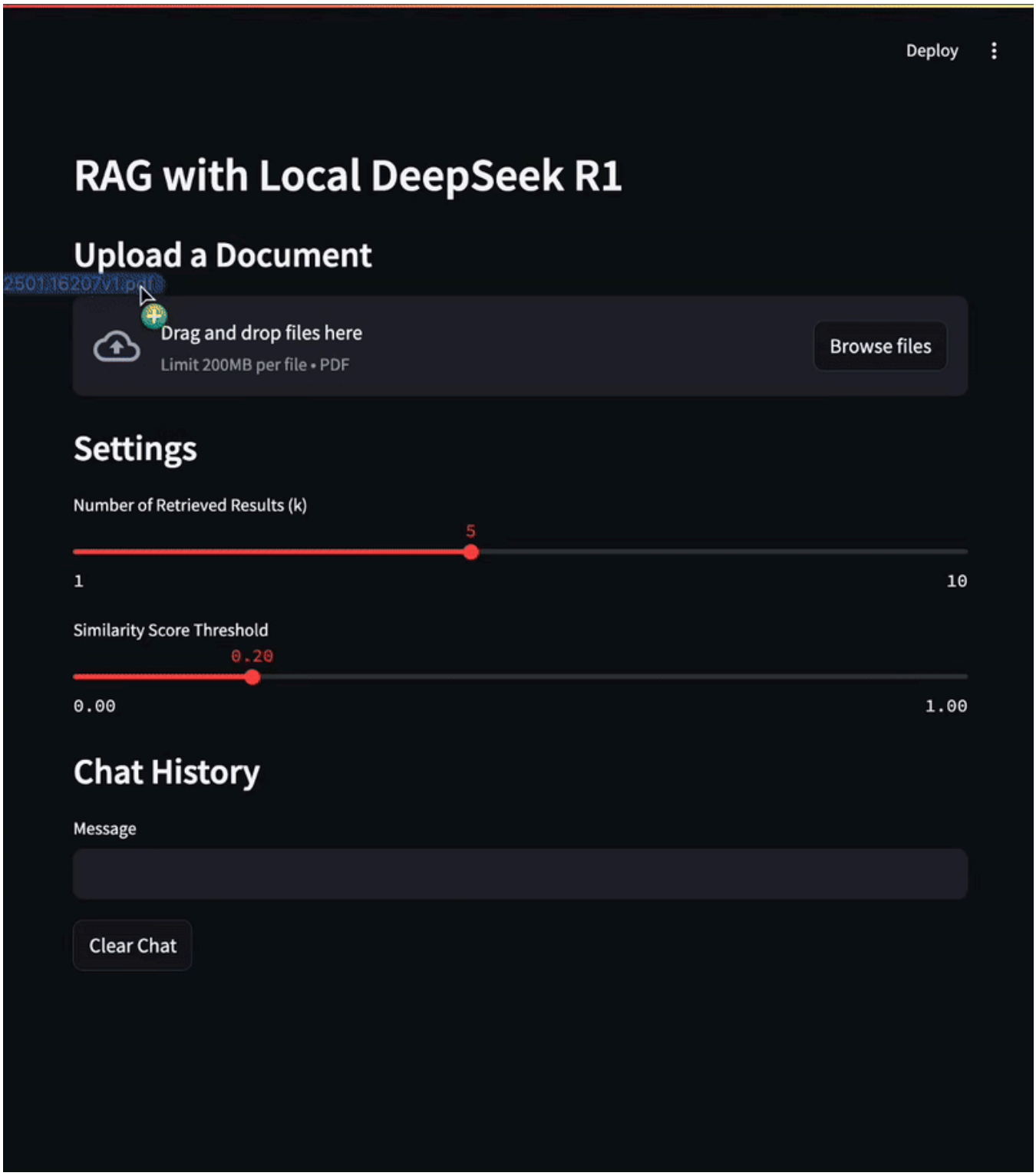
In this guide you'll learn how to build a **Retrieval-Augmented Generation (RAG)** system that processes PDFs locally using DeepSeek-R1, **LangChain**, **Ollama** and **Streamlit**. This step-by-step tutorial combines the modular power of **LangChain** with the privacy-first approach of **DeepSeek-R1**, offering a robust solution for handling technical, legal, and academic documents.



RAG Tech stack: DeepSeek-R1, Ollama, LangChain, and Streamlit

This project combines **LangChain**, an AI framework for RAG workflows, with **Ollama** for **DeepSeek-r1** local deployment and **Streamlit** for a user interface. The result is an AI assistant that can **ingest PDFs locally** and **answer questions** with precision and speed.

In this demonstration we are going to use a **DeepSeek-r1** distilled model of 7B parameters, but if you have more computational power I would recommend other DeepSeek-r1 distilled models.



RAG in Action: Upload a Document, Ask a Question, and Get a Response!

Why Choose a Private RAG Solution?

Cloud-based AI solutions are powerful but often come with challenges like **privacy risks** and **recurring costs**. By leveraging LangChain's modular framework, you can create a **local RAG solution** with numerous benefits:

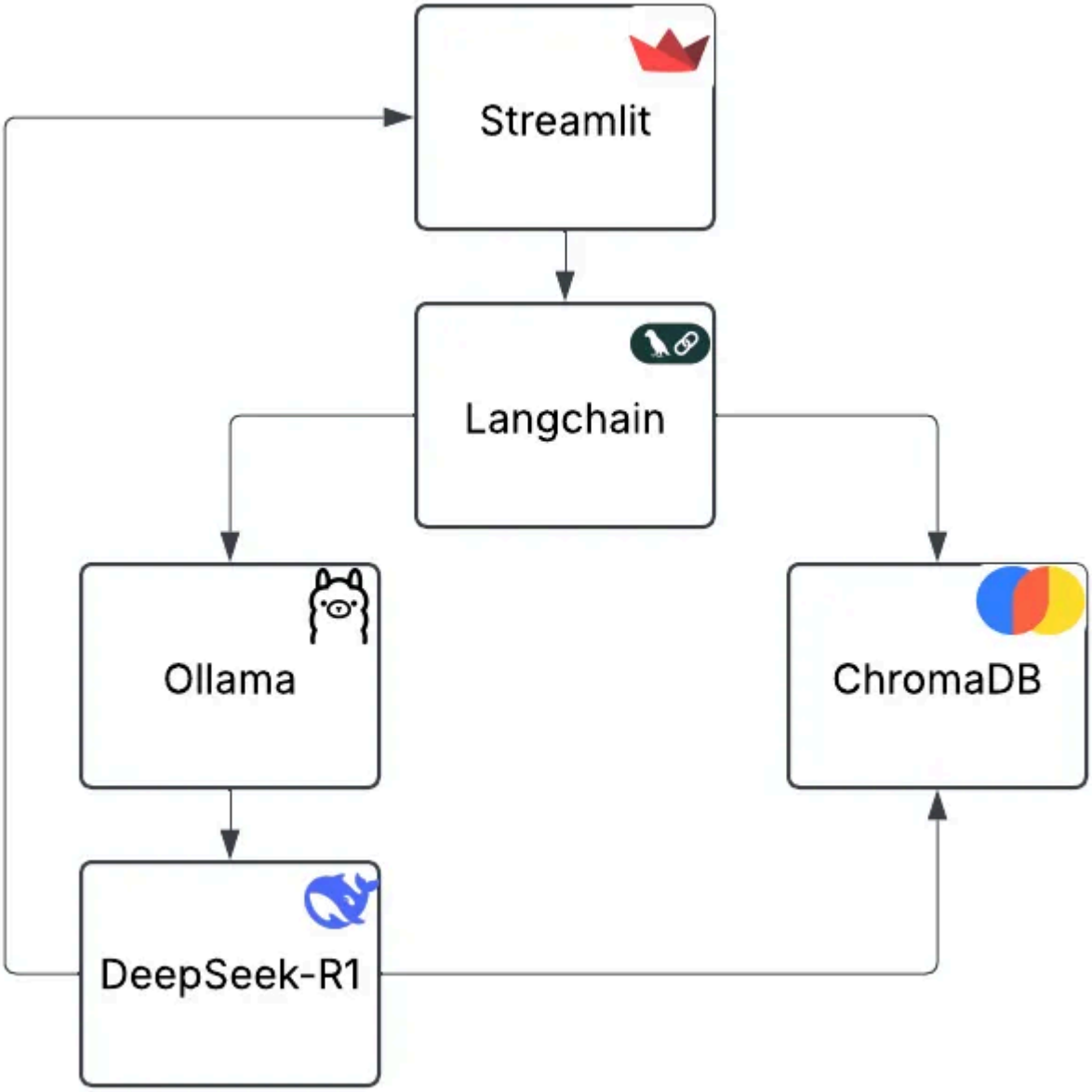
- **Data Privacy**: All operations happen locally, your data never leaves your machine.

- **Cost Efficiency**: No expensive API subscriptions, this solution is free and open-source.
- **Customizability**: LangChain's flexibility allows you to fine-tune the document retrieval and response generation pipeline.
- **Powerful AI**: Integrate with **DeepSeek-R1**, a reasoning model optimized for problem-solving and technical tasks.

Tools and Technologies: LangChain, DeepSeek-R1, Ollama, ChromaDB and Streamlit

This project is composed by:

- **LangChain**: The core framework for the RAG pipeline, enabling integration of document loaders, vector stores, and LLMs. It allows for modular and scalable AI workflows tailored to your specific needs.
- **DeepSeek-R1**: A reasoning LLM designed for coding, problem-solving, and technical tasks. Available in multiple distilled sizes for local deployment with Ollama.
- **Ollama**: A CLI tool that simplifies deploying and managing local LLMs and embedding models, such as DeepSeek-R1 and mxbai-embed-large.
- **ChromaDB**: A vector database that stores and retrieves document embeddings for similarity-based queries.
- **Streamlit**: A Python library for building web interfaces, making your RAG application user-friendly and accessible.



RAG Architecture Diagram

Building the RAG Pipeline: Step-by-Step Guide

Here's how you can set up your local ChatPDF solution:

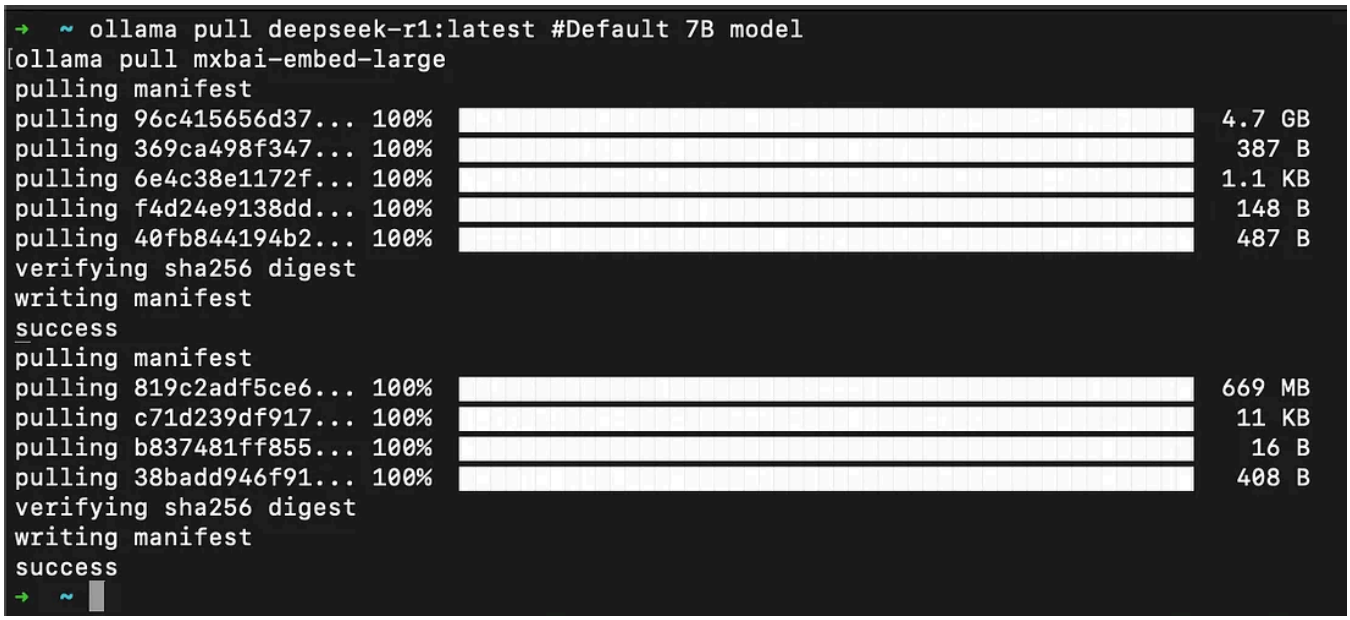
1. Install Prerequisites

Make sure you have Python 3.8+ and **Ollama** installed. Run the following commands:

```
curl -fsSL https://ollama.com/install.sh | sh
ollama -v  # Verify installation
```

Download the required AI models:

```
ollama pull deepseek-r1:latest # Default 7B model
ollama pull mxbai-embed-large  # Embeddings model
```



Downloading Deepseek-r1:7B and mxbai-embed-large

2. Set Up the Project

Clone the repository and set up a virtual environment:

```
git clone https://github.com/paquino11/chatpdf-rag-deepseek-r1.git
cd chatpdf-rag-deepseek-r1
python3 -m venv venv
source venv/bin/activate
```

Install the dependencies:

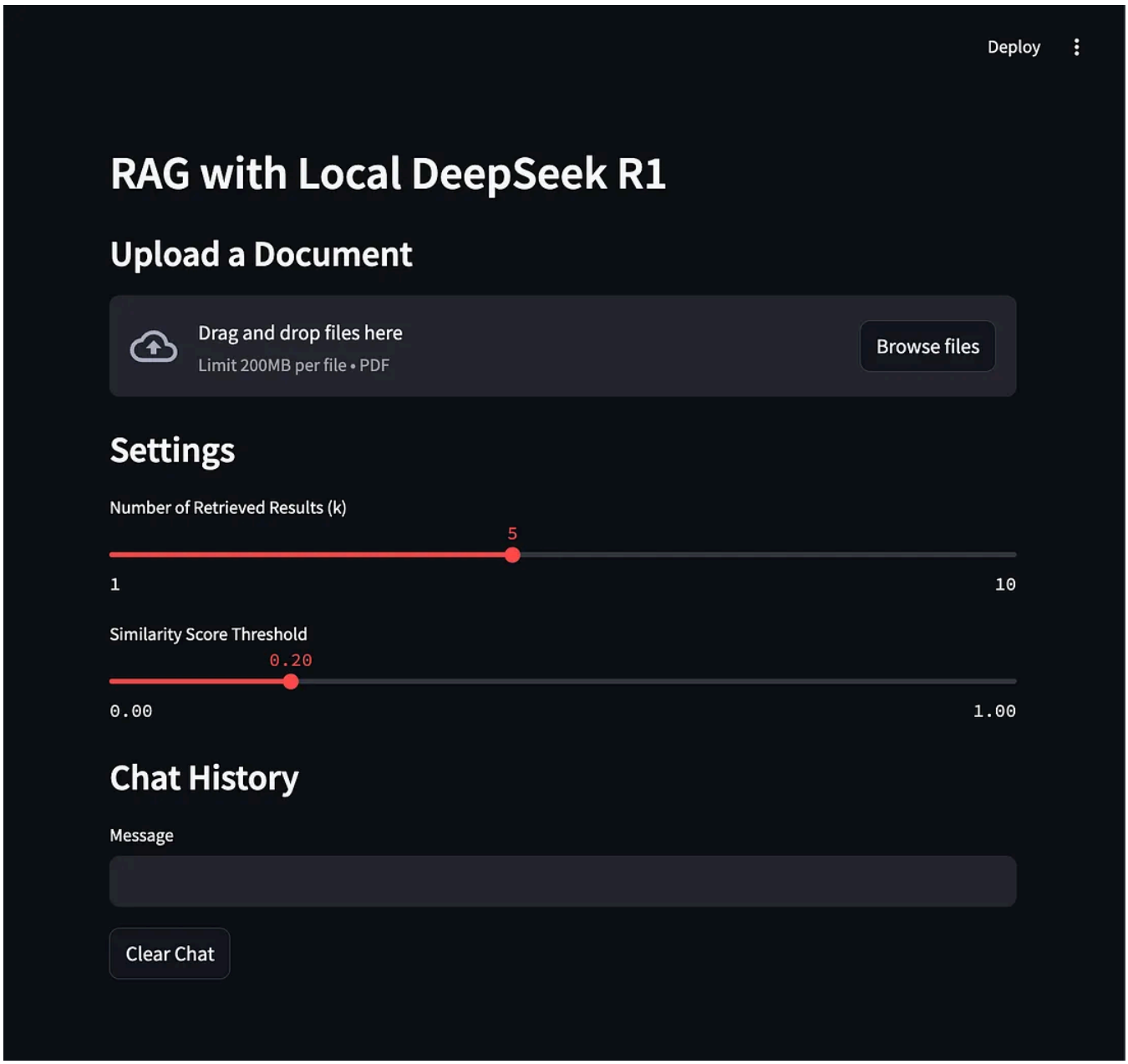
```
pip install -r requirements.txt
```

3. Run the Application

Start the Streamlit app:

```
streamlit run app.py
```

Access the app in your browser at `http://localhost:8501`. Upload your PDFs, adjust retrieval settings and start asking questions.



Streamlit UI for Local RAG with DeepSeek-R1

Building a RAG Pipeline with DeepSeek-R1, Ollama, LangChain and ChromaDB

This project uses LangChain to manage the entire RAG workflow:

1. **PDF Ingestion with LangChain:**

- PDFs are read and split into chunks using LangChain's **PyPDFLoader** and **RecursiveCharacterTextSplitter**.
- Chunks are embedded into vector representations with **OllamaEmbeddings**.

2. Document Retrieval with ChromaDB:

- LangChain's integration with **ChromaDB** enables fast, similarity-based retrieval of relevant document chunks.
- Customize the number of results (`k`) and similarity threshold (`score_threshold`) for better control.

3. Response Generation with DeepSeek-R1:

- Retrieved document chunks are passed to **DeepSeek-R1**, which generates concise and accurate answers.
- LangChain's **ChatPromptTemplate** ensures the AI responds in a user-friendly format.

Customizing Retrieval Settings for Optimal Results

LangChain makes it easy to tweak retrieval settings for optimal performance:

`k` : Number of Retrieved Results

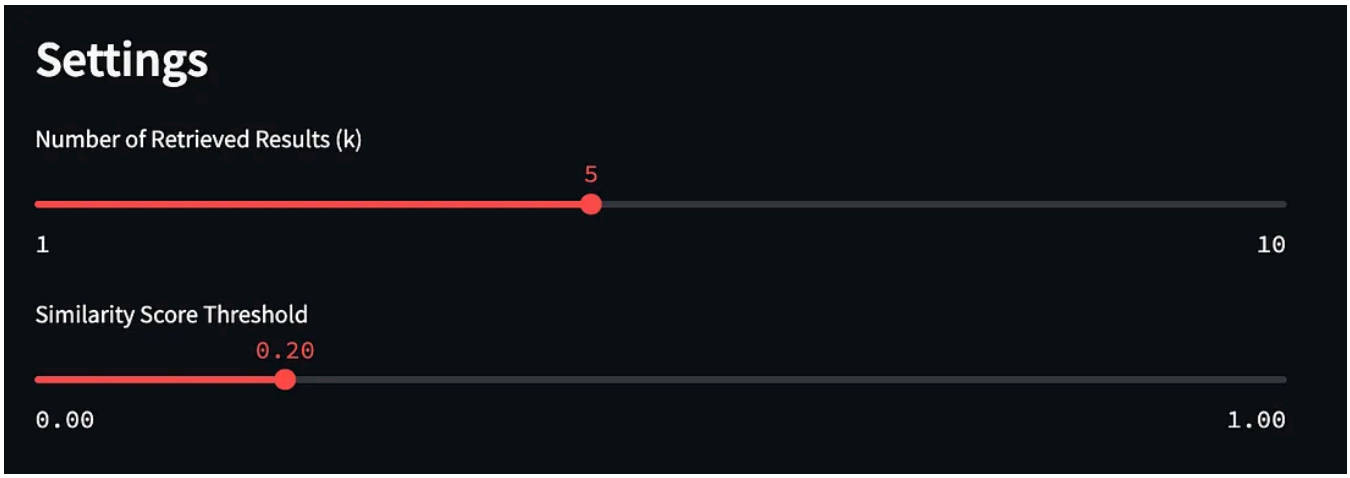
Controls how many document chunks are used in the response.

- Higher `k` : More context, slower response.
- Lower `k` : Less context, faster response.

`score_threshold` : Similarity Cutoff

Filters retrieved results based on relevance.

- Higher threshold: Only highly relevant chunks are retrieved.
- Lower threshold: Broader context but less precise.



Adjusting Retrieval Settings in the RAG App: Number of Retrieved Results (k) and Similarity Score Threshold.

Use Cases and Testing Your RAG Application

Here are some examples to test the app:

Test PDFs:

- *Finance*: Analyze financial reports and extract actionable insights.
- *Healthcare*: Summarize research papers or medical guidelines.
- *Education*: Extract summaries or key points from e-books and academic papers.

Sample Questions:

- "What are the key features of this Python library?"
- "What does Section 5 of this contract discuss?"
- "Summarize Chapter 2 of this e-book."

Conclusion

By combining **LangChain**, **DeepSeek-R1**, and **ChromaDB**, you can create a RAG system that prioritizes privacy, flexibility, and cost efficiency. This local RAG solution is great for analyzing technical documents, legal texts, and more, without relying on cloud-based tools.

References:

https://ollama.com/
https://python.langchain.com/docs/tutorials/rag/
https://docs.streamlit.io/
https://api-docs.deepseek.com/

Llm AI Deepseek Artificial Intelligence Machine Learning

Follow

Published in GoPenAI

2K Followers · Last published 2 days ago

Where the ChatGPT community comes together to share insights and stories.

Follow

Written by Pedro Aquino

223 Followers · 8 Following

Hi! I'm Pedro. I love building AI fullstack apps and sharing my learnings on AI, Automation, SaaS, and AI Frameworks.

Responses (6)

Vijay Agrawal

What are your thoughts?

Letizia
Feb 3

Hi Pedro, thank you for this!

If I might, how can this be integrated with the open webui interface, so that we can create multiple RAGs and on webui we can create custom models that use a specific RAG? Hope it's not asking too much! Thanks!

2 1 reply Reply

Kasun Herath
Feb 2

Thanks, it works!

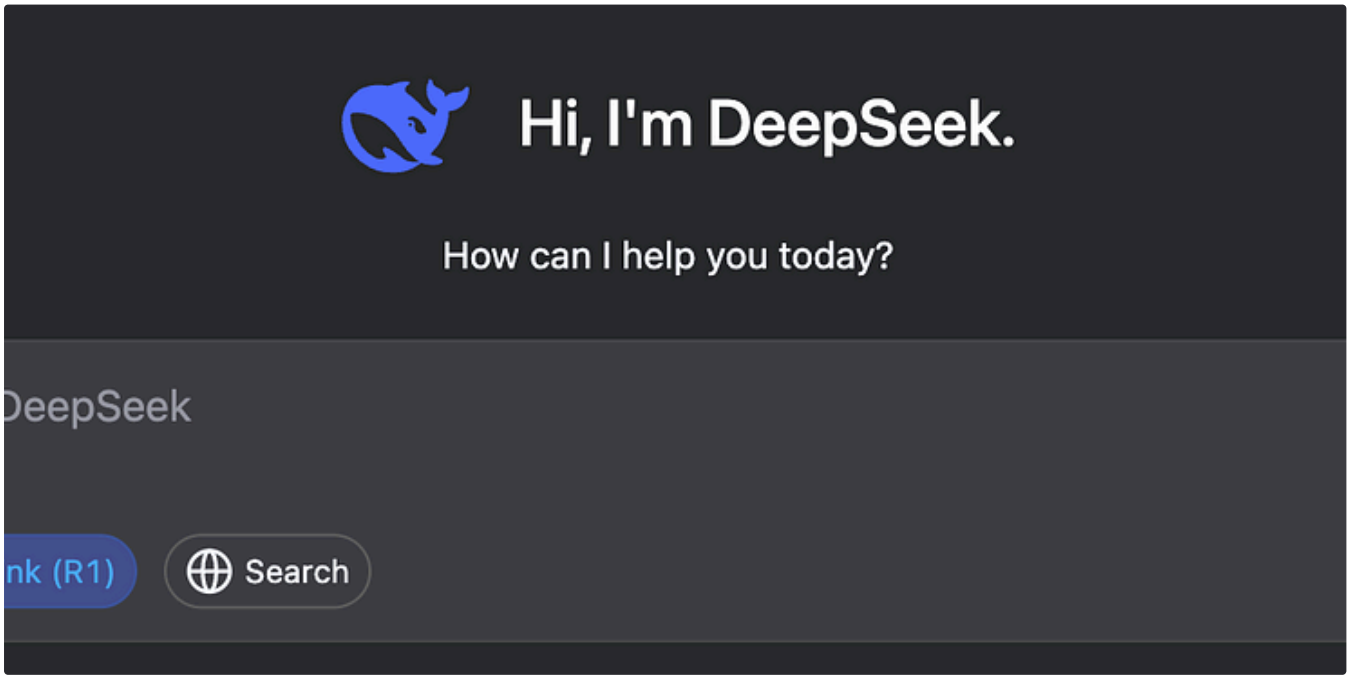
2 1 reply Reply

Bsaguirre
Jan 31

Works perfectly, nicely done!

1 1 reply Reply

More from Pedro Aquino and GoPenAI



Pedro Aquino

How to Install and Use DeepSeek-R1: A Free and Privacy-First Alternative to OpenAI (Save...

Learn how to install DeepSeek-R1 locally for coding and logical problem-solving, no monthly fees, no data leaks.

Jan 26 378 7

In GoPenAI by kirouane Ayoub

Contextual Embeddings with ModernBERT : A Hands-On Guide to Fine-Tuning ModernBERT Embed

In this blog post, we'll dive into ModernBERT, a significant upgrade to traditional BERT models. While we previously explored the concept...

Jan 31 208 4

In GoPenAI by Paras Madan

Building a Multi-Agent System for writing a Book: Crew AI - Tutorial + Colab Notebook

AI agents are becoming the next big thing, and multi-agent systems (MAS) are a perfect example of their power. They simplify complex...

Jan 20 120 4

Pedro Aquino

How to Use MCP Tools on Claude Desktop App and Automate Your Daily Tasks

Model Context Protocol (MCP) is a new standard for secure connection between AI assistants, such as Claude, and systems where data exists...

Dec 8, 2024 90 7

See all from Pedro Aquino

See all from GoPenAI

Recommended from Medium

In Generative AI by Crank Lee

DeepSeek+ Local Knowledge Base: Impressively Powerful

Today, I will share the deployment of Deepseek + local knowledge base.

Feb 8 300 4

In Towards AI by Lorentz Yeung

Comparing DeepSeek-R1 Models: 32B vs 70B vs R1

DeepSeek has made waves in the AI world. They offer multiple models at the same time, so which one should we choose?

Feb 4 35 4

Lists

Natural Language Processing
1963 stories · 1607 saves

Predictive Modeling w/ Python
20 stories · 1846 saves

AI Regulation
6 stories · 702 saves

Generative AI Recommended Reading
52 stories · 1673 saves

CyberRaya

Document RAG using Deepseek R1

Introduction

Feb 1 11

Sudarshan Koirala

Reasoning RAG (100% Private)

RAG with Reason Is ALL You Need

Feb 20

Ayush Gupta

Building a LLM Agent to Directly Interact with a Database

Large Language Models (LLMs) have revolutionized the way we interact with data and build intelligent applications. In this guide, I will...

Feb 22 60

In Level Up Coding by Pavan Belagatti

What's the Best PDF Extractor for RAG? I Tried LlamaParse, Unstructured and Vectorize

If you're building retrieval augmented generation (RAG) applications, you will eventually need to work with documents that are in PDF form.