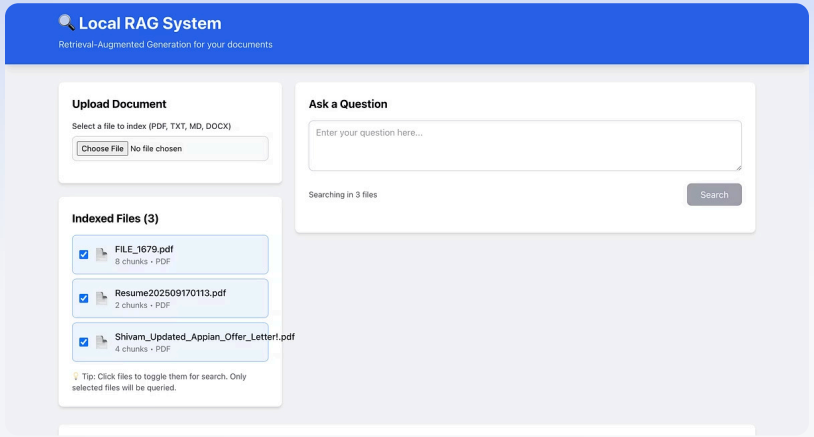
Local RAG Vector Search System

Semantic Document Search with AI-Generated Answers

Hey everyone! We are Shivam and Rui, and this is our project for CS 4774 – a local RAG Vector Search System. You can check out all the code on GitHub: https://github.com/RuiZhang-kwf8/uva-machine-learning-25f-projects



The Problem & Our RAG Solution

1

The Problem

- Traditional keyword search often misses the true meaning behind your query.
- Large Language Models (LLMs) can't answer questions using your specific documents.

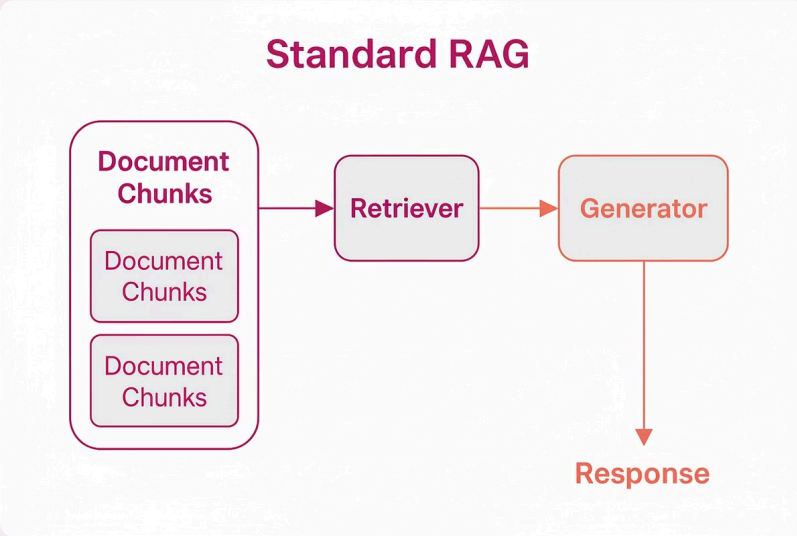
2

Our RAG Solution

RAG (Retrieval-Augmented Generation) solves this by:

- Retrieving relevant context from your private documents.
- Feeding that context to an LLM for accurate, grounded answers.

This project brings RAG locally, ensuring privacy and giving you AI-powered answers with sources directly from your uploaded documents.



System Architecture: All Local, All Private

Our system is built with a modern stack, entirely local for privacy and control.

Frontend: React

User-friendly interface for uploading documents and asking questions.

Backend: FastAPI

Handles all the heavy lifting, connecting the frontend to our AI magic.

Vector DB: FAISS

Super-fast similarity search for finding relevant document chunks.

LLM: Ollama (LLaMA 3.2)

Generates human-like answers, all running right on your machine.

This setup means no cloud dependencies, keeping your data secure and your interactions private. We use sentence-transformers for embeddings.

RAG Deep Dive: Document Indexing

Here's how we prepare your documents for intelligent retrieval:

1. Document Upload & Extraction

Upload PDFs, TXT files, or Markdown. We extract all the text.

2. Chunking

We break the text into smaller pieces (500 tokens) with some overlap (100 tokens).

3. Embedding Generation

Each chunk is converted into a 384-dimensional vector, capturing its semantic meaning.

4. FAISS Storage

These embeddings are stored in our FAISS vector database, ready for quick search.

Why chunking? It helps fit content into LLM context limits and ensures precise retrieval. Why overlap? To maintain context continuity across chunks.

RAG Deep Dive: Query Pipeline

Once your documents are indexed, here's how we answer your questions:

1. User Question

1

You ask a question about your documents.

2. Query Embedding

2

Your question is converted into a vector in the same space as the document chunks.

3. Similarity Search

3

FAISS quickly finds the top-5 most semantically similar chunks (k-NN search).

4. Prompt Construction

4

We build a special prompt for the LLM, including system instructions, the retrieved chunks, and your original question.

5. LLM Answer

5

The LLM generates an answer, strictly based on the provided context.

6. Answer + Citations

6

You get a grounded answer with clear source citations from your documents!

The key here is that the LLM's answer is grounded in facts from your documents, not hallucinated.

Key Design Decisions & ML Concepts

We made deliberate choices to build this system, highlighting several core ML principles.

Our Choices

Local Embeddings

No API costs and full data privacy.

FAISS Vector Search

Fast k-NN searches, proven for large datasets.

Fixed Chunking

Simple and predictable, making context management easier.

Local LLM (Ollama)

Free, private, and fully offline capable AI generation.

ML Concepts Demonstrated

Transfer Learning

Leveraging pre-trained models for embeddings and generation.

Semantic Similarity

Using embeddings to understand meaning, not just keywords.

Prompt Engineering

Crafting effective prompts for RAG to guide the LLM.

Trade-offs

Balancing quality vs. speed, cost vs. convenience in system design.

Demo & Performance Snapshot

Let's take a quick look at the system in action and its performance metrics.

Demo Walkthrough

Imagine uploading a research paper, then asking, "What are the main findings of this study?" The system will:

- Show you the uploaded document being processed.
- Display the retrieved chunks with their similarity scores.
- Present the AI-generated answer, complete with citations back to your document.

Performance Metrics

2.5s

Embedding 1000 sentences

100ms

Search 10K vectors

2-5s

LLM response time

This system works brilliantly for document Q&A and efficient knowledge retrieval.

Limitations, Future Work & Conclusion

No project is perfect, but we have a clear path forward!

1

Current Limitations

- Single-stage retrieval (could add re-ranking for precision).
- Fixed chunk size (adaptive chunking would be better).
- No formal evaluation metrics yet.

2

Future Improvements

- Hybrid search (vector + keyword) for even better results.
- Streaming responses for a smoother user experience.
- Develop an evaluation suite (Recall@k, MRR).

Conclusion: A Robust RAG System

- ✓ Functional end-to-end RAG system.
- ✓ Demonstrates practical ML integration.
- ✓ Production-quality code & documentation.

Ready to explore? Find the full project on GitHub: https://github.com/shivam-agra/local-rag-system

Thank You!

Questions? Thoughts? Let's discuss!

Your feedback is invaluable as we continue to improve this project. Don't hesitate to reach out!