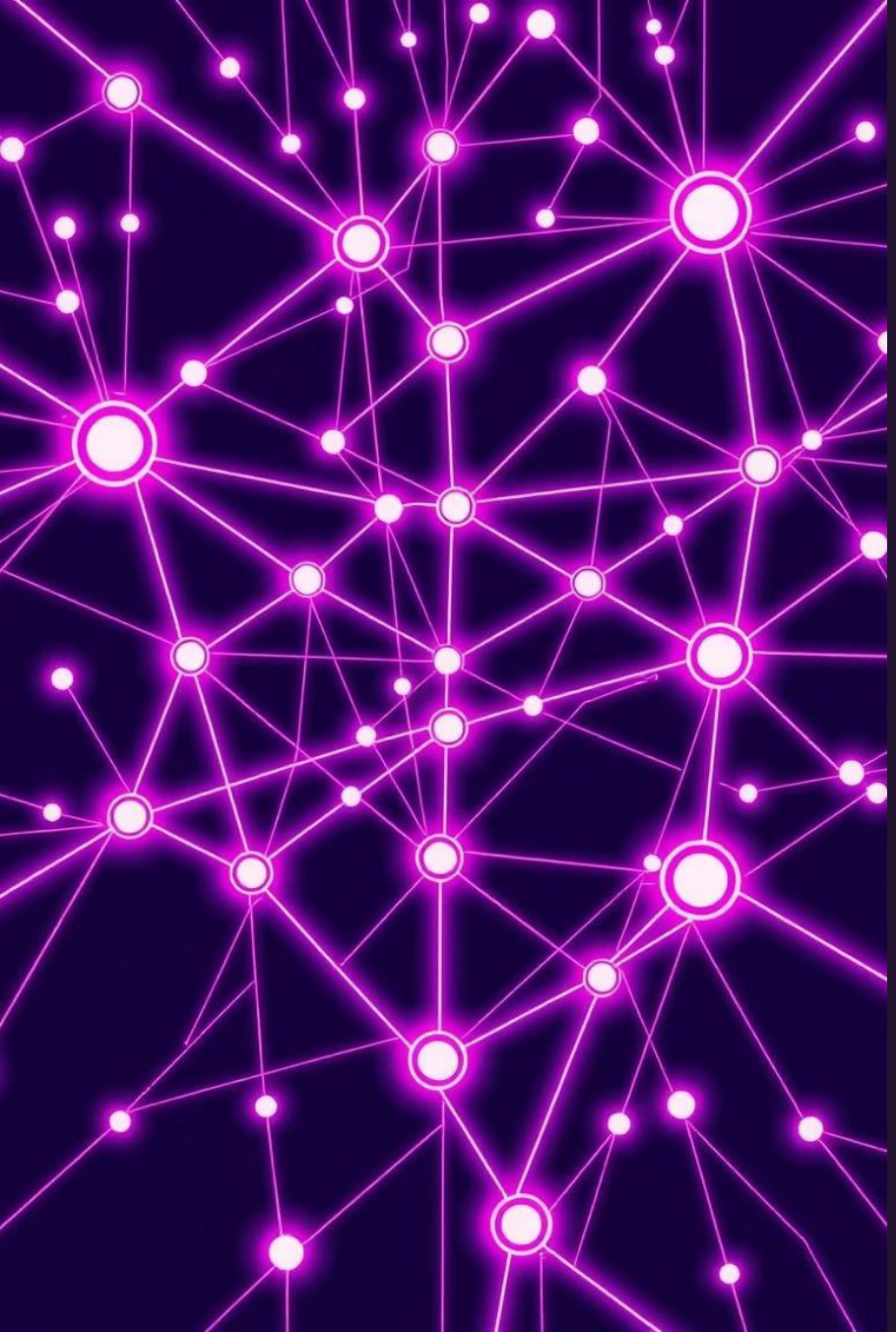


RAG Lab

Retrieval-Augmented Generation with Mistral and FAISS

An intelligent system combining vector search and language model inference to generate context-aware answers from PDF documents.

The Challenge: Beyond Static LLMs

Language models have inherent knowledge cutoffs and lack access to domain-specific information. RAG systems solve this by augmenting LLMs with real-time retrieval capabilities, enabling accurate, contextual answers grounded in your actual data.

The Problem

LLMs cannot access external documents or real-time data without RAG.

The Solution

Retrieve relevant context first, then generate answers grounded in that context.

System Architecture

The RAG pipeline orchestrates six interconnected stages, from raw document input to intelligent answer generation:

Document Ingestion

Extract text from PDF files to establish your knowledge base.

Text Chunking

Split content into overlapping semantic chunks to preserve context.

Embedding

Convert chunks into numerical vectors using SentenceTransformer models.

Vector Indexing

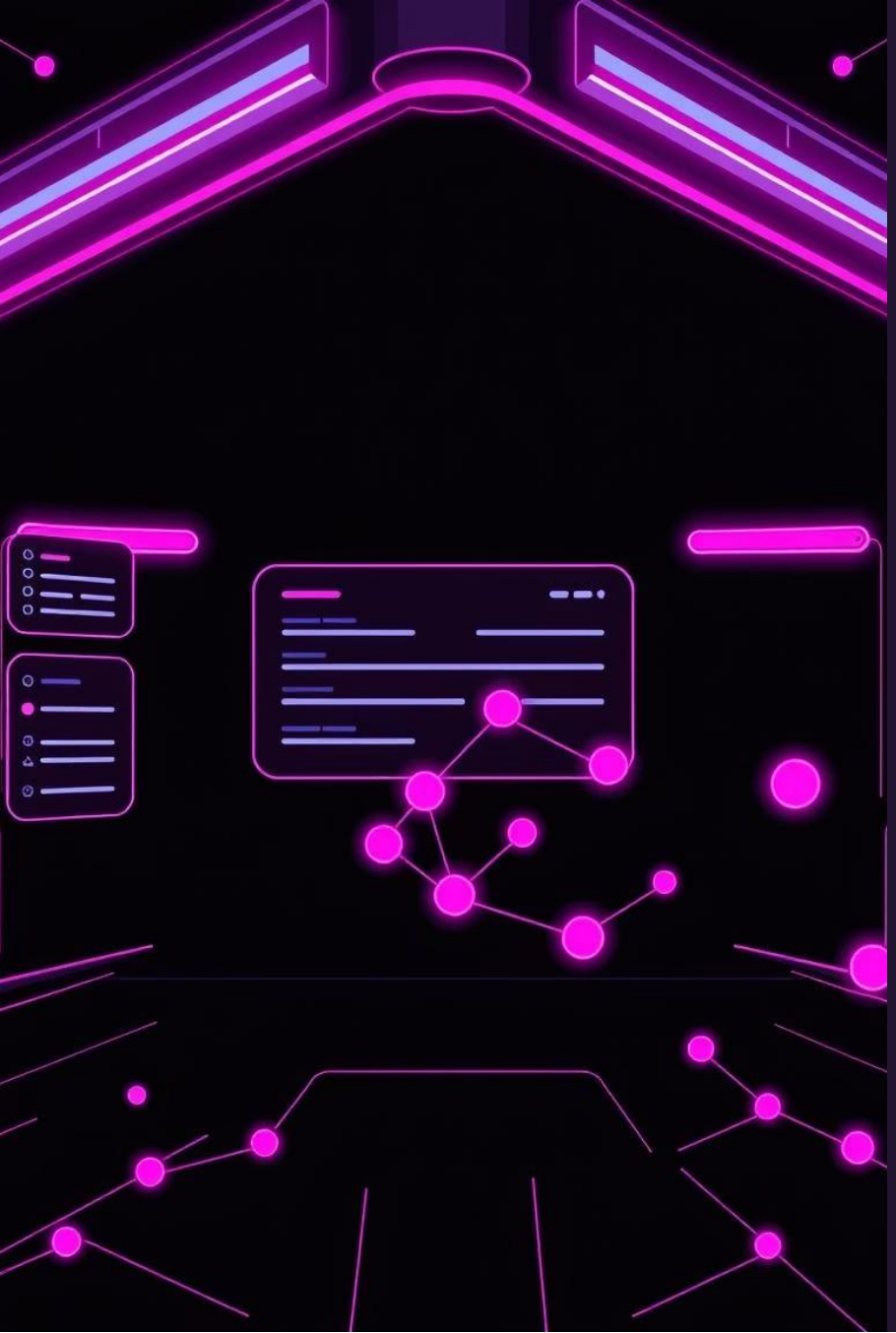
Store embeddings in FAISS for efficient similarity search.

Retrieval

Find most relevant chunks matching the user's query embedding.

Generation

Feed retrieved context to Mistral for final answer synthesis.



Core Components

Input Processing

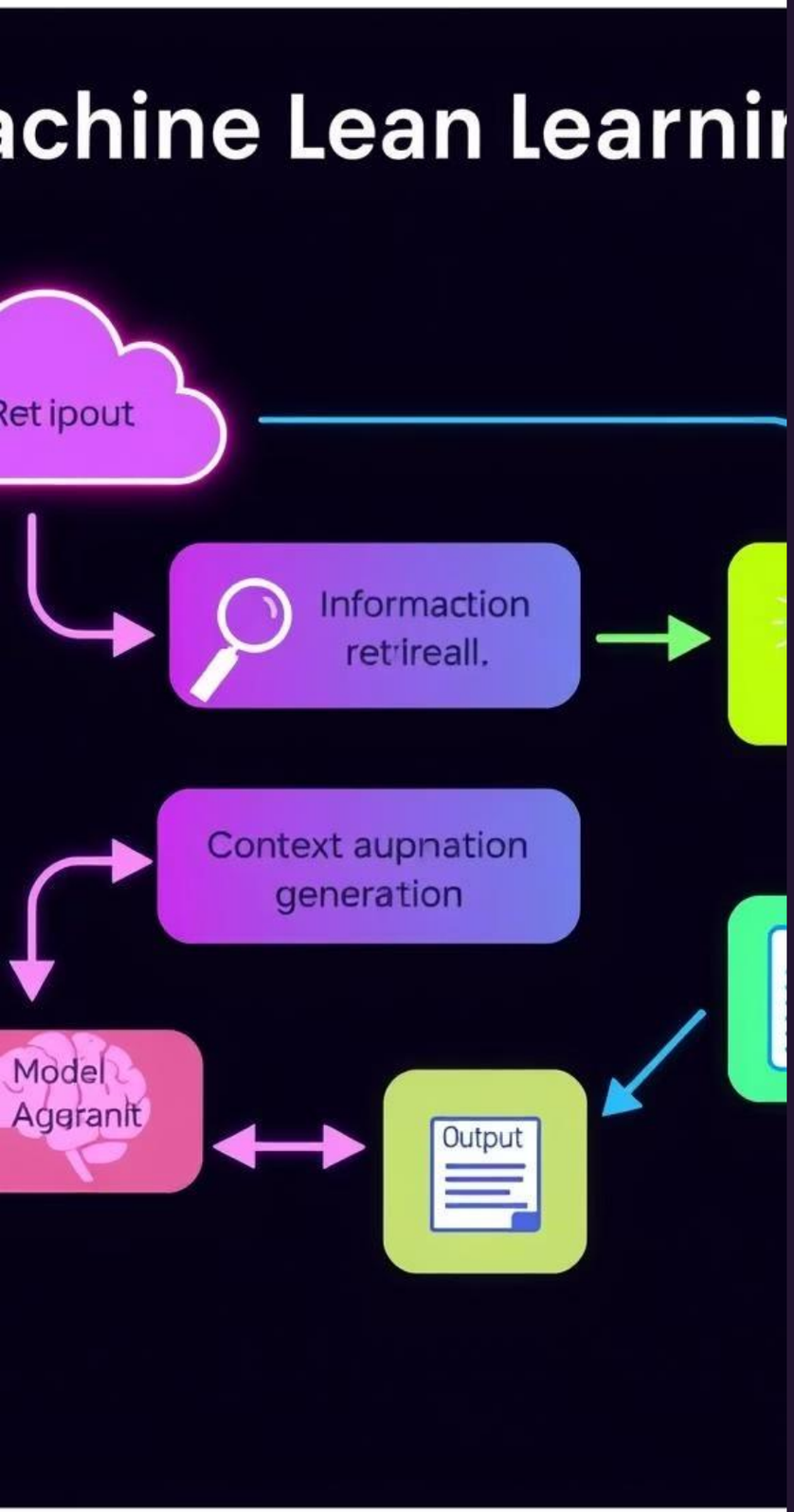
Document Ingestion: PyPDF2 extracts text from PDF files, creating the raw knowledge base for your system.

Text Chunking: Content is divided into small overlapping chunks, ensuring semantic context continuity between retrieval boundaries.

Semantic Representation

Text Embedding: SentenceTransformer models transform text chunks into high-dimensional vectors capturing semantic meaning.

Vector Indexing: FAISS efficiently stores and searches embeddings, enabling millisecond-scale similarity queries across large datasets.



The Retrieval-Generation Pipeline

01

Query Embedding

User question is embedded using the same SentenceTransformer model, producing a query vector in the same semantic space as indexed chunks.

02

Similarity Search

FAISS compares the query vector against all stored embeddings, returning the top-k most similar chunks ranked by cosine distance.

03

Context Assembly

Retrieved chunks are aggregated and formatted as contextual prompt augmentation for the language model.

04

LLM Generation

Mistral-Nemo-Instruct generates a coherent, factually grounded answer using the retrieved context and original question.

Technical Stack

Language Models

Transformers library loads and runs Mistral-Nemo-Instruct for inference.

Embeddings

Sentence-Transformers generates semantic representations of text chunks.

Vector Search

FAISS provides GPU-accelerated similarity search across embedding indexes.

Data Processing

PyPDF2, NumPy, and Pandas handle document extraction and numerical operations.

Real-World Applications

Academic Support

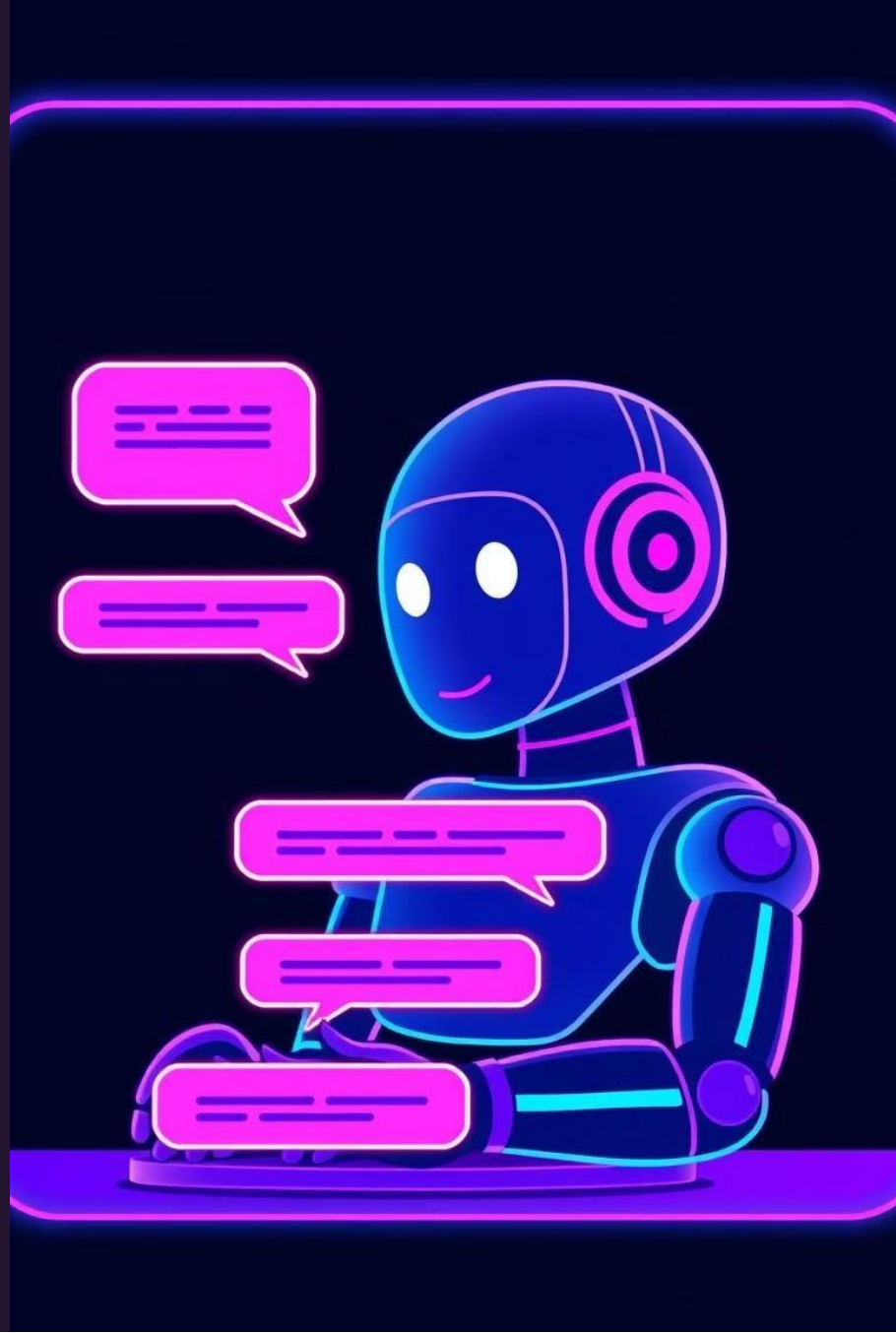
Answer university-related queries grounded in course materials and institutional documents.

Information Extraction

Rapidly extract key information, summaries, and facts from large document collections.

Custom Assistants

Build domain-specific chatbots and Q&A systems grounded in your proprietary data sources.



Scaling Forward

The RAG foundation enables powerful extensibility:

1 Multi-Document Support

Ingest and index embeddings from multiple PDFs simultaneously, building unified knowledge bases across document collections.

2 Persistent Storage

Store embeddings and metadata in vector databases (Pinecone, Weaviate, Milvus) for production-scale retrieval at scale.

3 Web Interfaces

Deploy Streamlit or Flask applications exposing RAG systems via interactive dashboards and APIs.

4 Performance Optimisation

Implement query caching, experiment with alternative embedding models, and benchmark retrieval accuracy against your domain.