RAG from Scratch

Short Description: Learn how to build a Retrieval-Augmented Generation system from the ground up, understanding every component of the pipeline so you can create, debug, and extend RAG systems with confidence.

About the Course

Course Overview

This course will teach you how to build a Retrieval-Augmented Generation system from scratch. RAG is a technique that connects large language models to external knowledge sources, allowing them to answer questions about data they were never trained on. Rather than relying on libraries that hide the implementation details, this course walks you through every stage of the pipeline so you understand exactly how and why each part works.

We will start by building the mental model: why LLMs hallucinate, what parametric and non-parametric knowledge mean, and when RAG is the right choice over fine-tuning or prompt stuffing. From there, we will move through document preparation, chunking strategies, vector embeddings, and cosine similarity search, implementing each step from scratch before moving to the next.

Finally, we will put everything together by building a complete, working RAG system applied to a real use case. By the end of the course, you will have a functioning pipeline and a clear understanding of every design decision that went into it.

Course Objectives

By the end of this course, you will be able to:

Explain why RAG exists and how it differs from fine-tuning and other LLM techniques.

Load, clean, and prepare documents with metadata for use in a retrieval pipeline.

Design and implement chunking strategies and evaluate their impact on retrieval quality.

Generate vector embeddings and implement cosine similarity search from scratch.

Build a RAG prompt template that grounds the LLM in retrieved context and prevents hallucinations.

Assemble a complete end-to-end RAG system and apply it to a real use case.

Prerequisites

This course is designed for developers with basic Python knowledge who want to understand how RAG systems work from the inside out. No prior experience with embeddings, vector search, or large language model frameworks is required.

Syllabus

About This Course

Welcome to RAG from Scratch. Large language models are powerful, but they do not know your data. They cannot answer questions about your internal documents, company policies, or domain-specific knowledge that falls outside their training set. Retrieval-Augmented Generation, or RAG, is the technique that bridges this gap by combining a retrieval step with LLM generation to produce grounded, accurate answers.

This course takes a ground-up approach. Rather than relying on high-level libraries that hide the details, you will build each component of the RAG pipeline from scratch. You will start with the theory behind why RAG exists, move through document preparation, chunking, embeddings, and similarity search, and finish by building a complete, working RAG system. By the end of the course, you will understand every decision that goes into a RAG pipeline and have the skills to build, adapt, and improve one in your own projects.

Learning Outcomes

By the end of this course, you will be able to:

- Explain why RAG exists, how it differs from fine-tuning, and when it is the right choice.
- Trace the full RAG pipeline from a user query to a grounded answer.
- Load, clean, and prepare documents with metadata for use in a retrieval system.
- Design chunking strategies and evaluate the trade-offs of size and overlap.
- Generate vector embeddings and implement cosine similarity search from scratch.
- Build a RAG prompt template that grounds the LLM in retrieved context and handles missing answers.
- Assemble a complete, end-to-end RAG system applied to a real use case.

Prerequisites

- Basic Python programming knowledge.
- Familiarity with how large language models work at a conceptual level.
- No prior experience with RAG, embeddings, or vector search is required.

Course Syllabus

Chapter	Content	Project / Assignment

Introduction	Course overview. What RAG is and why it matters. The problem with LLMs on private data.	
Chapter 1 Setting the Mental Model	The limits of LLMs. Parametric vs non-parametric knowledge. Hallucinations and stale knowledge. What RAG is and what it is not. RAG vs fine-tuning, prompt stuffing, and tool calling. Real-world use cases.	Identify a real-world problem where RAG is a better choice than fine-tuning.
Chapter 2 What We Are Building and Why	The full RAG pipeline in one diagram. What each stage does. Common misconceptions about RAG systems. Walk through a working RAG demo and trace the query flow step by step.	Trace a sample query through each stage of the pipeline.
Chapter 3 Preparing the Knowledge Base	Choosing a dataset. Text extraction basics. Cleaning and normalization. Why document structure matters. Storing documents with IDs and metadata.	Prepare a small document set of 10 to 20 documents and identify useful metadata fields.
Chapter 4 Chunking	Why chunking is required. Chunk size trade-offs. Overlap and its impact on retrieval. When chunking fails. Comparison of large vs small chunks and overlap vs no overlap.	Experiment with two chunking strategies and document the differences.
Chapter 5 Embeddings and Similarity Search	What embeddings represent. Choosing an embedding model. Cosine similarity explained intuitively. Generating embeddings for chunks. Implementing similarity search manually.	Write a basic similarity search function and test it with multiple queries.
Chapter 6 Prompting the LLM with Retrieved Context	RAG prompt structure. Separating instructions from context. Preventing hallucinations. Handling cases where the answer is not found. Testing queries with and without retrieval.	Design a prompt template that forces grounded answers only.

Capstone Complete RAG System in Action	Combine all pipeline stages into a single working RAG system. Apply the system to a real use case end to end.	Build and run a complete RAG pipeline on a document set of your choice.