M06 — Retrieval-Augmented Generation (RAG) Lab

Lab — Generative AI Boot Camp for Developers — Exported 2026-09-02

Time: ~25 min | **Day 2 PM** (shared with M07)

2026-09-02

Learning Objectives

Build a RAG-based knowledge assistant. Ingest documents, query with grounding, and compare RAG vs. non-RAG responses.

2026-09-02

Concept Review: RAG in Practice

The RAG Pipeline

```
1. INGEST: Documents → chunks → embeddings → vector DB
2. QUERY: User question → embed → search vector DB → retrieve top-K chunks
3. AUGMENT: Build prompt with retrieved chunks as context
4. GENERATE: Send augmented prompt to LLM → grounded response
```

Key Components

Component	What It Does	Options
Embedding Model	Converts text to vectors	Chromadb built-in (onnx, free), OpenAI embeddings
Vector Database	Stores and searches embeddings	ChromaDB (lightweight, pip install), Pinecone (cloud)
Chunking Strategy	How you split documents	Fixed-size (200-500 tokens), sentence-based, semantic
Top-K Retrieval	How many chunks to retrieve	Start with 3-5; more = broader context but higher token cost

2026-09-02

Why RAG Instead of Fine-Tuning?

RAG	Fine-Tuning
Dynamic — docs update instantly	Static — retrain when data changes
No model training needed	Requires training data and compute
Works with any LLM	Tied to a specific model
Provides citations (retrieved chunks)	No built-in provenance
Cheaper and faster to implement	Better for consistent style/tone shifts

Common Failure Modes

- **Retrieval misses relevant docs** → Chunks too large or too small; try different chunk size
- **Irrelevant chunks confuse the LLM** → Lower top-K; add relevance threshold
- **Context window overflow** → Too many chunks; reduce top-K or chunk size
- **Stale data** → Vector DB not re-indexed; schedule regular ingestion

---

Prerequisites

- LLM API access (for embeddings and generation)
- ChromaDB: `pip install chromadb`
- Sample documents provided in `workshop/m06/docs/`

> **Using OpenCode or VS Code Chat:** Both work for this lab. OpenCode is good for executing the Python scripts. VS Code Chat is good for explaining concepts as you go.

2026-09-02

Exercise: Build a RAG Knowledge Assistant

Step 1 — Ingest Documents (10 min)

Sample documents are in `workshop/m06/docs/`. Create an ingestion script that: 1. Reads each document 2. Splits into chunks (~300 tokens each, 50-token overlap) 3. Generates embeddings (uses chromadb built-in model) 4. Stores in ChromaDB

Use **OpenCode** to generate the ingestion script:

```
opencode run "Write a Python script that reads all .txt files from workshop/m06/docs/,
splits each into chunks of ~300 words with 50-word overlap, generates embeddings using
chromadb (built-in, install if needed), and stores in a ChromaDB
collection named 'knowledge_base'."
```

Step 2 — Query with RAG (10 min)

Implement a `query_rag(question)` function: 1. Embed the question 2. Search ChromaDB for top 3 chunks 3. Build an augmented prompt: "Using the following information: [chunks]... Answer: [question]" 4. Call LLM with the augmented prompt 5. Return the grounded response

```
opencode run "Read the ingestion script. Add a query_rag(question) function that:
embeds the question, searches ChromaDB for the top 3 most similar chunks, builds a
prompt that includes those chunks as context, and returns the LLM's response. Use
the same LLM provider you configured in M01."
```

2026-09-02

Step 3 — Compare: RAG vs. No RAG (5 min)

Ask the same question twice — once directly to the LLM (no RAG) and once through your `query_rag()` function. Compare the answers.

Which would you trust? Which provides citations/references?

Deliverable

Working `query_rag()` function + before/after comparison of RAG vs. non-RAG answers.

2026-09-02

Troubleshooting

Symptom	Fix
ChromaDB import error	`pip install chromadb`
Embedding API error	Check API key; chromadb has a built-in free embedding model
Retrieval returns wrong docs	Adjust chunk size; try 200 or 500 instead of 300
Augmented prompt too long	Reduce top-K from 5 to 3; reduce chunk size
No documents to ingest	Create a few sample .txt files with domain info you know well

2026-09-02

Hints

- **Start small:** 2-3 documents, 5-10 chunks total. Verify the pipeline works before scaling.
- **Test with a question you know the answer to** — this lets you validate retrieval quality.
- **The comparison (Step 3) is the most important part** — don't skip it.

2026-09-02