

lab

April 3, 2026

1 Lab 09 — Advanced RAG

Upgrade your RAG pipeline with hybrid search, reranking, and evaluation.

Exercises: 1. Hybrid search with BM25 + reciprocal rank fusion 2. Cross-encoder reranking 3. LLM-as-judge faithfulness evaluation

```
[ ]: # Setup
import os
import json
import numpy as np
from pathlib import Path
from dotenv import load_dotenv
import anthropic
import chromadb
from sentence_transformers import SentenceTransformer, CrossEncoder
from rank_bm25 import BM25Okapi

load_dotenv()
client = anthropic.Anthropic()
MODEL = "claude-sonnet-4-20250514"
embed_model = SentenceTransformer("all-MiniLM-L6-v2")
chroma_client = chromadb.Client()
print("Setup complete.")
```

```
[ ]: # Load documents (reuse from Lesson 08)
docs_dir = Path("../lesson08-rag-basics/demo/sample_docs")
doc_files = sorted(docs_dir.glob("*.txt"))
documents = [f.read_text(encoding="utf-8") for f in doc_files]
doc_names = [f.stem for f in doc_files]
print(f"Loaded {len(documents)} documents")

# Create ChromaDB collection
embeddings = embed_model.encode(documents).tolist()
collection = chroma_client.create_collection("python_docs")
collection.add(
    ids=doc_names,
    documents=documents,
```

```

        embeddings=embeddings,
    )
    print(f"ChromaDB collection: {collection.count()} docs")

```

1.1 Exercise 1 — Hybrid Search with BM25

Implement BM25 search, dense search, and reciprocal rank fusion.

```

[ ]: # TODO: Create BM25 index
# TODO: Implement bm25_search(query, k=5) -> list of doc indices
# TODO: Implement dense_search(query, collection, k=5) -> list of doc indices
# TODO: Implement reciprocal_rank_fusion(ranked_lists, k=60) -> merged ranked_
    ↪ list
# TODO: Implement hybrid_search(query, k=5) -> list of doc indices

raise NotImplementedError("Exercise 1: implement hybrid search")

```

```

[ ]: # Test Exercise 1
print("Keyword query:")
for idx in hybrid_search("lru_cache decorator", k=3):
    print(f"  {doc_names[idx]}")

print("\nSemantic query:")
for idx in hybrid_search("how to measure code performance", k=3):
    print(f"  {doc_names[idx]}")

```

1.2 Exercise 2 — Cross-Encoder Reranking

Score candidate documents with a cross-encoder and take the top results.

```

[ ]: # TODO: Load cross-encoder model
# TODO: Implement rerank(query, doc_indices, top_k=3) -> list of doc indices
# TODO: Implement advanced_search(query, retrieve_k=10, final_k=3) -> list of_
    ↪ doc indices

raise NotImplementedError("Exercise 2: implement reranking")

```

```

[ ]: # Test Exercise 2
query = "How do I create a custom dictionary with default values?"
print("Advanced search results:")
for idx in advanced_search(query, retrieve_k=10, final_k=3):
    print(f"  {doc_names[idx]}: {documents[idx][:60]}...")

```

1.3 Exercise 3 — LLM-as-Judge Evaluation

Compare naive vs advanced pipelines using faithfulness scoring.

```
[ ]: # TODO: Define 5 test questions
      # TODO: Implement judge_faithfulness(question, context, answer) -> int (1-5)
      # TODO: For each question, get answers from naive and advanced pipelines
      # TODO: Score both and build comparison table

      raise NotImplementedError("Exercise 3: implement evaluation")
```