RAG (Retrieval-Augmented Generation) Learning Roadmap

1. Prerequisites

- Python programming
- Basic NLP understanding
- Transformers & LLMs (e.g., GPT, BERT)
- REST APIs & JSON

2. Basics of Retrieval & Generation

Retrieval:

- TF-IDF, BM25
- Vector embeddings & semantic search

Tools: scikit-learn, faiss, sentence-transformers

Generation:

- Learn LLMs (GPT, T5)
- Prompt engineering

Tools: OpenAI API, Hugging Face

3. RAG Architecture

Typical flow:

1. Input query
2. Retriever fetches documents
3. Generator answers using retrieved data

4. Tools & Frameworks

- Hugging Face Transformers
- Haystack
- LangChain
- LlamaIndex
- ChromaDB, Weaviate, Pinecone

RAG (Retrieval-Augmented Generation) Learning Roadmap

5. First RAG System

- Small doc corpus + embeddings (e.g., MiniLM)
- Store in faiss/Chroma
- Generate answers with OpenAI GPT-4 or LLaMA

6. Chunking & Optimization

- Optimal chunking (token overlap)
- Compare embedding models
- Use filters/metadata in search

7. Evaluation & Fine-Tuning

- Evaluate with ragas
- Tackle hallucinations
- Fine-tune retriever/ranker

8. Advanced Topics

- Multi-hop RAG
- Tool-augmented RAG
- Hybrid (BM25 + dense)
- Secure/private RAG
- Cost-optimization

9. Projects to Practice

1. Doc QA for PDFs/Notion
2. Slackbot with LangChain
3. RAG + Memory agent
4. Enterprise RAG with access control

10. Keep Up to Date

RAG (Retrieval-Augmented Generation) Learning Roadmap

- GitHub: LangChain, LlamaIndex, Haystack
- Newsletters: Latent Space
- Papers: RAG (2020), InstructRAG (2023)