Training Summary Document

Retrieval-Augmented Generation (RAG) and LangChain: Concepts, Design, and Implementation

Trainer Name: Samrawit Gebremaryam
Date: Dec 6, 2025

Introduction to RAG and LangChain

This training introduces participants to **Retrieval-Augmented Generation (RAG)** and its practical implementation using **LangChain**, a leading framework for building LLM-based applications.

The RAG approach enhances Large Language Models by giving them access to **external knowledge sources**, ensuring answers are factual, up-to-date, and grounded in real documents. Instead of relying entirely on the internal memory of an LLM—which may hallucinate or lack recent information—RAG retrieves relevant information and fuses it with generative reasoning.

LangChain provides modular components for building intelligent LLM-driven systems:

- Document ingestion (loading & splitting)
- Embedding generation
- Vector search and retrieval
- Prompt orchestration
- Evaluation and optimization

Through this training, participants will gain the conceptual and hands-on skills needed to create a functional RAG system, implement document chatbots, and design retrieval pipelines tailored to real-world applications such as knowledge assistants, search tools, workflow help systems, and enterprise intelligence solutions.

Training Objective

The objectives of this training are to:

1. **Understand the principles of Retrieval-Augmented Generation (RAG)**
    - Why LLMs need external knowledge retrieval

- RAG architecture and workflow
- Use cases across industries

2. **Explore LangChain and its core components**

    - Document loaders
    - Text splitters
    - Embeddings and vector databases
    - Retrievers and LLM chains
    - Pipeline orchestration

3. **Design and build a complete RAG system**

    - Create embeddings
    - Populate a vector store
    - Build retrievers
    - Connect everything into a working RAG chain

4. **Apply RAG to real documents or datasets**

    - PDFs, DOCX, text logs, research papers, or product documents
    - Querying the document
    - Improving retrieval accuracy

Training Content

1. Motivation: Why RAG and Why LangChain?

- LLM limitations (hallucinations, outdated knowledge)
- Importance of augmenting models with real data
- How LangChain simplifies LLM application development
- Advantages over custom Python scripts:

    - Faster prototyping
    - Ready-made modules
    - Reliable integrations
    - Reusable pipelines

2. Core Concepts of LangChain

a. Document Loading

- Import PDFs, DOCs, web pages, GitHub files
- Multi-format ingestion

b. Text Splitting

- Creating optimal chunk sizes for retrieval
- Overlap management for semantic continuity

c. Embeddings

- Meaning representation of text
- Using OpenAI, HuggingFace, or local models

d. Vector Stores

- Storing embeddings in FAISS, ChromaDB, Pinecone

e. Retrievers

- Similarity search
- Multi-query retrieval
- Context filtering

f. RAG Chain Construction

- Retrieval → Augmentation → LLM Answering
- Prompt templates for structured output

3. Building a RAG System Using LangChain

Steps:

1. Load documents
2. Split into chunks
3. Generate embeddings
4. Store embeddings in a vector database
5. Create a retriever
6. Connect an LLM

7. Build the question-answering pipeline

Example:

- “Chat with Your PDF” system
- Querying course notes, company policies, or documentation

Learning Outcomes

By the end of the training, participants will be able to:

- Explain the RAG workflow and its importance
- Understand how LangChain components interact
- Build and customize a retrieval pipeline
- Load and preprocess documents efficiently
- Generate embeddings and work with vector databases
- Implement a complete RAG system in LangChain
- Deploy a document-based chatbot
- Integrate RAG into real products or internal systems

Training Task

Task Title:
Build a Document-Based RAG System Using LangChain

Objective:
Create a RAG system that can answer questions about a **private or domain-specific document** that the LLM cannot answer correctly without retrieval.

Instructions:

Document Selection:

- Choose a document containing private, internal, or domain-specific information (e.g., company policies, product manuals, internal research notes).
- The document must contain information that cannot be answered by the LLM alone.

Pipeline Implementation:

- Load the document using a LangChain document loader

- Split the document into chunks
- Generate embeddings for each chunk
- Store embeddings in a vector database (FAISS, Chroma, or Pinecone)
- Create a retriever to find relevant chunks
- Connect an LLM via LangChain to answer questions using retrieved content

Functionality:

- The system should accept user queries
- Answers must be based on retrieved data only
- Demonstrate that without retrieval, the LLM cannot answer correctly

Deliverables:

- Full RAG pipeline
- Uploaded document(s) used
- Explanation of architecture and design decisions
- Demonstration showing queries and retrieved answers

Evaluation Criteria

Conceptual Understanding:

- Understanding of RAG and LangChain components
- How RAG enables answering questions about private or unknown data
- How retrieval improves accuracy and prevents hallucinations

Presentation:
Participants will present their project to demonstrate:

- Understanding of RAG and LangChain components
- How retrieval improves answer accuracy
- Design and implementation of their RAG pipeline
- Design choices
- Examples of questions that require document retrieval
- Reflection on challenges and solutions

Assessment:

- Conceptual clarity

- Correct implementation of the RAG system
- Evidence that retrieval is essential for correct answers
- Ability to explain architecture and reasoning

Bonus Extensions (Optional):

- Add a GUI using Streamlit
- Use rerankers for improved accuracy
- Add metadata-aware filtering