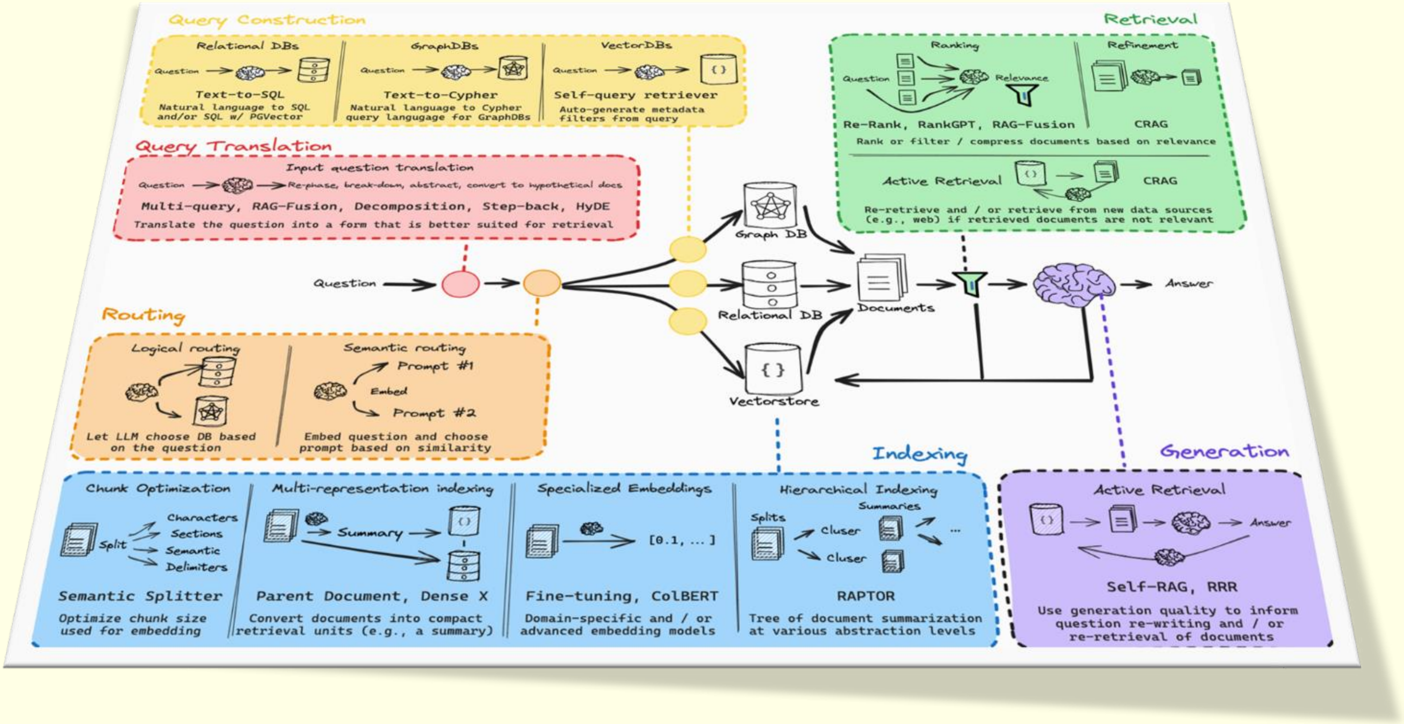
RAG Tutorial

(Retrieval Augmented Generation)

Setting Up Your Environment for LangChain RAG


Query Construction
Relational DBs
Question
Text-to-SQL
Natural language to SQL and/or SQL w/ PGVector
GraphDBs
Question
Text-to-Cypher
Natural language to Cypher query langugage for GraphDBs
VectorDBs
Question
Self-query retriever
Auto-generate metadata filters from query
Query Translation
Input question translation
Question
Re-phase, break-down, abstract, convert to hypothetical docs
Multi-query, RAG-Fusion, Decomposition, Step-back, HyDE
Translate the question into a form that is better suited for retrieval
Retrieval
Ranking
Question
Relevance
Re-Rank, RankGPT, RAG-Fusion
Refinement
CRAG
Rank or filter / compress documents based on relevance
Active Retrieval
CRAG
Re-retrieve and / or retrieve from new data sources (e.g., web) if retrieved documents are not relevant
Question
Graph DB
Relational DB
Vectorstore
Documents
Answer
Routing
Logical routing
Let LLM choose DB based on the question
Semantic routing
Prompt #1
Embed
Prompt #2
Embed question and choose prompt based on similarity
Indexing
Chunk Optimization
Split
Characters
Sections
Semantic
Delimiters
Semantic Splitter
Optimize chunk size used for embedding
Multi-representation indexing
Summary
Parent Document, Dense X
Convert documents into compact retrieval units (e.g., a summary)
Specialized Embeddings
[0.1, ...]
Fine-tuning, ColBERT
Domain-specific and / or advanced embedding models
Hierarchical Indexing
Summaries
Splits
Cluser
Cluser
RAPTOR
Tree of document summarization at various abstraction levels
Generation
Active Retrieval
Answer
Self-RAG, RRR
Use generation quality to inform question re-writing and / or re-retrieval of documents

Day 2 of 7

Table of Contents

1. Introduction

Welcome to Day 2 of our LangChain RAG tutorial series. Today, we'll focus on setting up your environment to work with LangChain, ensuring you have all the necessary tools and dependencies in place. We'll cover the use of Jupyter Notebooks, installation of required packages, setting up LangSmith for tracing, and provide a preview of our first application: building an app to answer questions about the content of a website.

2. Installation

We'll cover the use of Google Collab Notebooks, installation of required packages, setting up LangSmith for tracing, and provide a preview of our first application: building an app to answer questions about the content of a website.

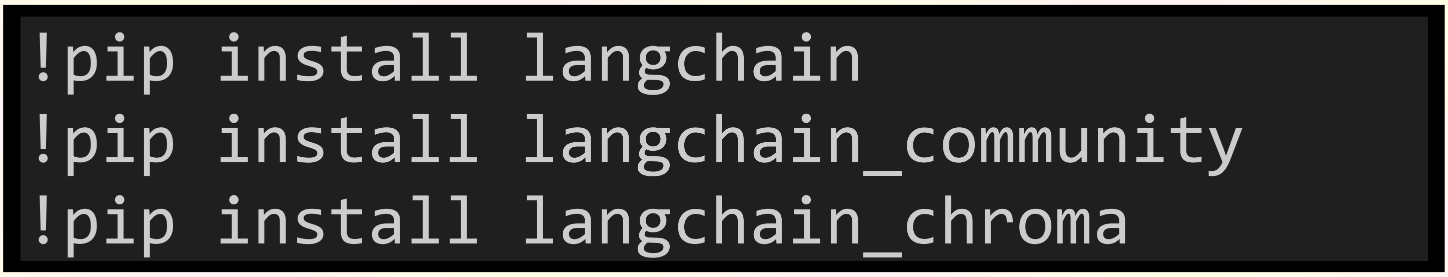
2.1 Google Collab Jupyter Notebook

Google Collab notebooks are ideal for learning and working with large language models (LLMs) and LangChain systems. They offer an interactive environment where you can experiment with code, debug issues, and visualize outputs.

2.2 Install the required dependencies

The complete sequence from raw data to the final answer generally includes:

Before diving into LangChain, we need to install the required dependencies. LangChain relies on several packages, and we recommend using pip for managing these installations.



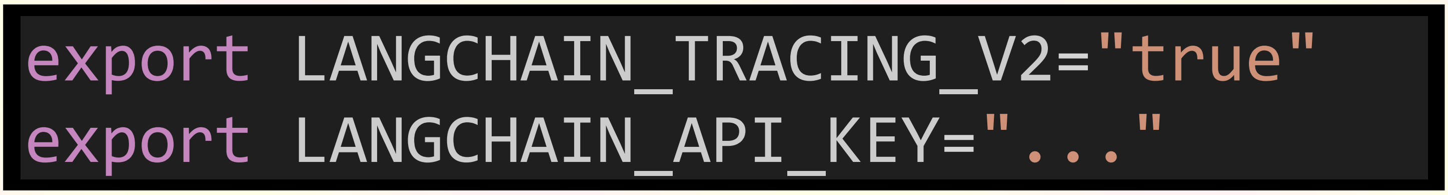
```
!pip install langchain
!pip install langchain_community
!pip install langchain_chroma
```

2.3 LangSmith

As you build more complex applications with LangChain, inspecting and debugging the multiple steps and LLM calls becomes crucial. LangSmith provides tools for logging and tracing these operations.

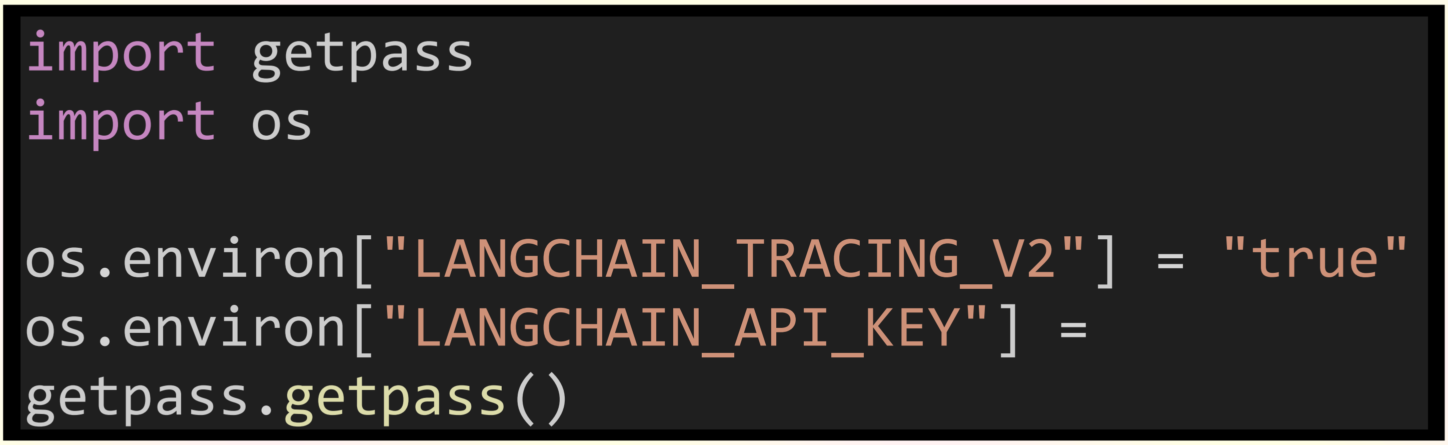
First, sign up for LangSmith **here**.

Then, set your environment variables to enable logging:



```
export LANGCHAIN_TRACING_V2="true"
export LANGCHAIN_API_KEY="..."
```

In a Google Collab notebook, you can set these variables as follows:



```python
import getpass
import os

os.environ["LANGCHAIN_TRACING_V2"] = "true"
os.environ["LANGCHAIN_API_KEY"] = getpass.getpass()
```

3. Preview: Building a Question-Answering App

In this Google Collab Notebook, we'll build an application that answers questions about the content of a specific website. For this example, we'll use the "LLM Powered Autonomous Agents" blog post by Lilian Weng. We’ll create a simple indexing pipeline and Retrieval Augmented Generation (RAG) chain to handle this in approximately 20 lines of code.

Link of collab Notebook

4. Conclusion

Today, we covered the essential setup for working with LangChain, including using Google Collab Notebooks, installing necessary dependencies, setting up LangSmith for tracing, and a preview of our first RAG application. In the next session, we'll dive deeper into building and refining this application.

Stay tuned for the next tutorial in this series, where we will dive deeper into optimizing and scaling your RAG applications.