Analytics Vidhya

Recap of Legacy and LCEL Chains

Instructor

Dipanjan Sarkar

Head of Community & Principal AI Scientist at Analytics Vidhya

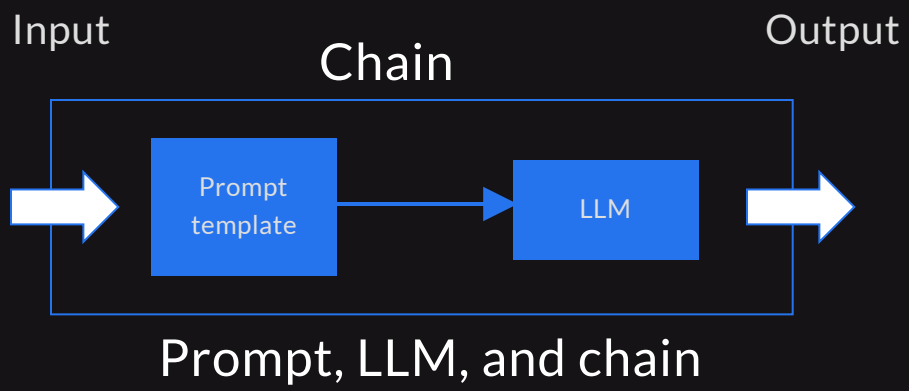
Google Developer Expert - ML & Cloud Champion Innovator

Published Author

Outline

- What is a LangChain Chain?
- LangChain Legacy Chains
- Popular Legacy Chains
- LangChain Expression Language - LCEL Chains
- Popular Built-in LCEL Chains
- LCEL Advantages

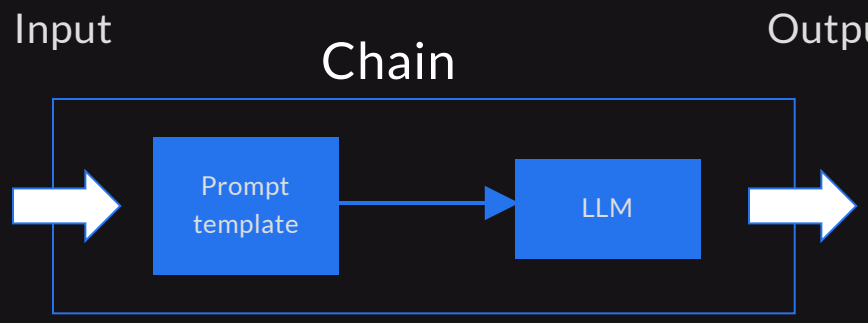
LangChain Legacy Chains



Prompt, LLM, and chain

- A Chain is a sequence or a pipeline of steps
- These steps are typically calls in LangChain
  - Calling an LLM
  - Calling a tool
  - Calling a retriever
  - Calling a preprocessing operation

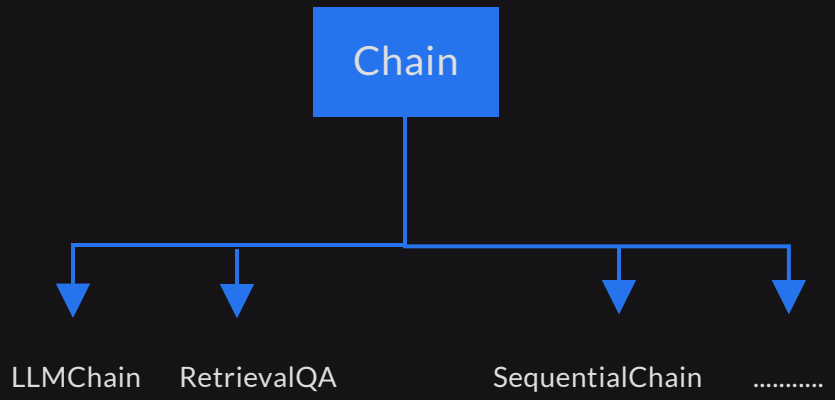
LangChain Legacy Chains



Prompt, LLM, and chain

- Prior to August 2023, we had a different syntax for creating LLM Chains
- LangChain now calls them as legacy Chains
- Newer syntax is built using LangChain Expression Language (LCEL)
- Not all legacy chains have been converted into LCEL variants

LangChain Legacy Chains

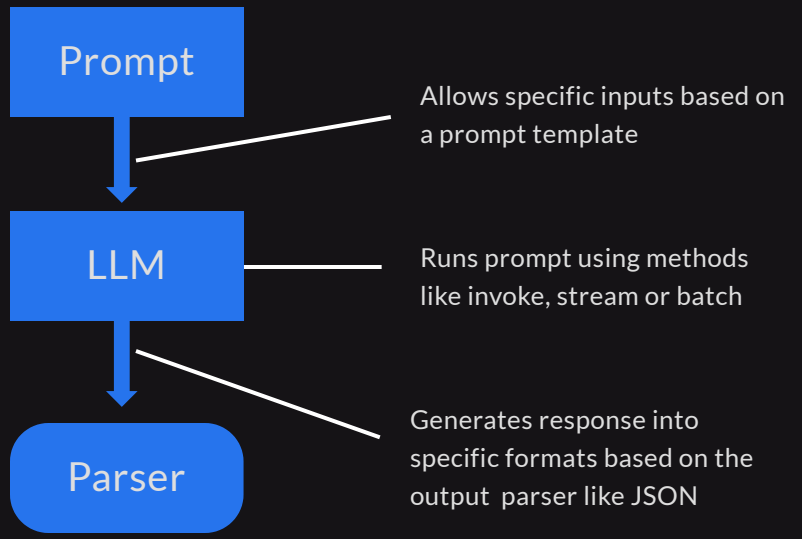


- Legacy Chains are constructed by subclassing from a legacy Chain class
- These do not use LCEL but have their own implementation logic
- Many of them are being deprecated so need to keep a check on LangChain new releases and docs
- Some of them are still used as there is no LCEL variant yet

Popular Legacy Chains

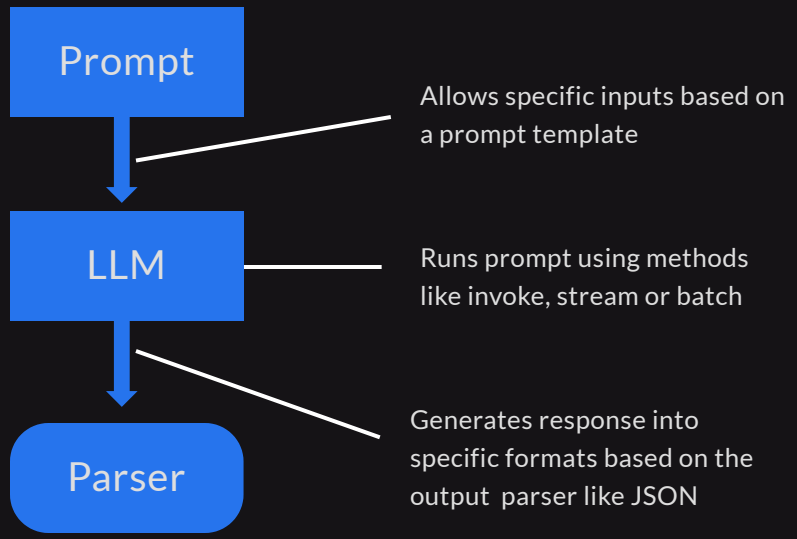
Chain Type	Description
LLMChain	Simplest chain to pass queries or prompts to LLMs and get responses
SequentialChain	Connect multiple chains sequentially, where the outputs of one chain feed directly into next
LLMRouterChain	A router chain that uses an LLM chain to perform routing prompts to specific chains based on the prompt and certain conditions
RetrievalQA	This chain is used to connect a vector database retriever to an LLM to build a RAG chain to get responses based on custom data
ConversationChain	Enables the chain to store past conversations in memory and have full conversation with the LLM
ConversationalRetrievalChain	Combination of conversational and RAG chains, enables you to get answers based on custom data and also have a conversation with the LLM

LangChain Expression Language - LCEL Chains



- LangChain Expression Language, or LCEL, is a declarative way to easily compose chains together
- It has quickly become the de-facto standard to build complex LLM pipelines or chains
- Recommended officially by LangChain when building LLM apps

LangChain Expression Language - LCEL Chains

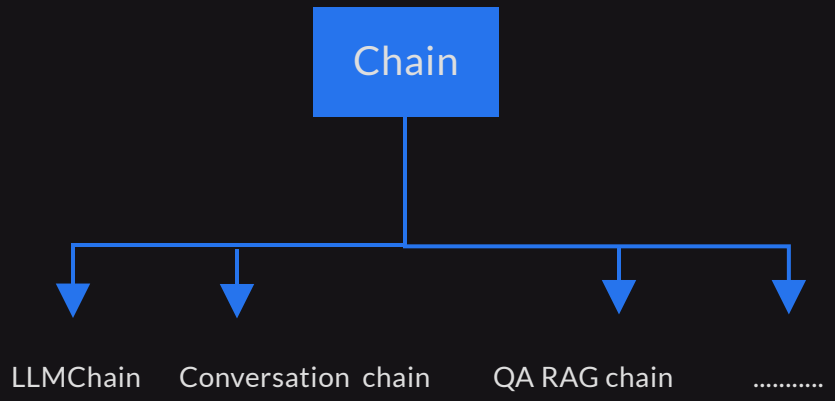


- In LCEL you can chain multiple steps using the overloaded vertical bar or pipe 'I' operator
- Example LCEL chain on the left can be created as:
  - `chain = Prompt | LLM | Parser`
  - This signifies the prompt flows into the LLM which generates the response which is formatted using the Parser rules

Popular Built-in LCEL Chains

Chain Type	Description
create_stuff_documents_chain	This chain takes a list of documents and formats them into a prompt, and passes that prompt to an LLM to generate a response
create_sql_query_chain	Create a chain that generates SQL queries for the given database from natural language
create_history_aware_retriever	This chain takes in the conversation history and then uses that to generate a rephrased search query if needed, which is passed to the underlying retriever to retrieve relevant documents
create_retrieval_chain	This is a standard RAG chain. It takes a user query and passes it to the retriever to fetch relevant context documents. The query and context are then passed to an LLM to generate a response

Custom LCEL Chains



- Any custom LCEL chain can be built by stacking runnables with the pipe '|' operator
- Popular chains we will explore include:
  - LLM Chain
  - Conversation Chain
  - QA RAG Chain

LCEL Advantages

Streaming support: LCEL chains enable you to get chunks of response tokens with low latency and stream the response live to the user

Async support and parallel execution: LCEL chains have support for async APIs to handle many concurrent requests in the same server. You can also execute certain steps in parallel if possible

Deployment, monitoring and observability: LCEL chains can be deployed easily with LangServe. They allow you to access the results of intermediate steps, log them using LangSmith for observability, debugging and monitoring

Input and Output Schemas: LCEL chains have capabilities to validate inputs and outputs based on specific schemas

Thank You

Analytics Vidhya