Analytics Vidhya

LangChain Expression Language (LCEL)

Instructor

Dipanjan Sarkar

Head of Community & Principal AI Scientist at Analytics Vidhya

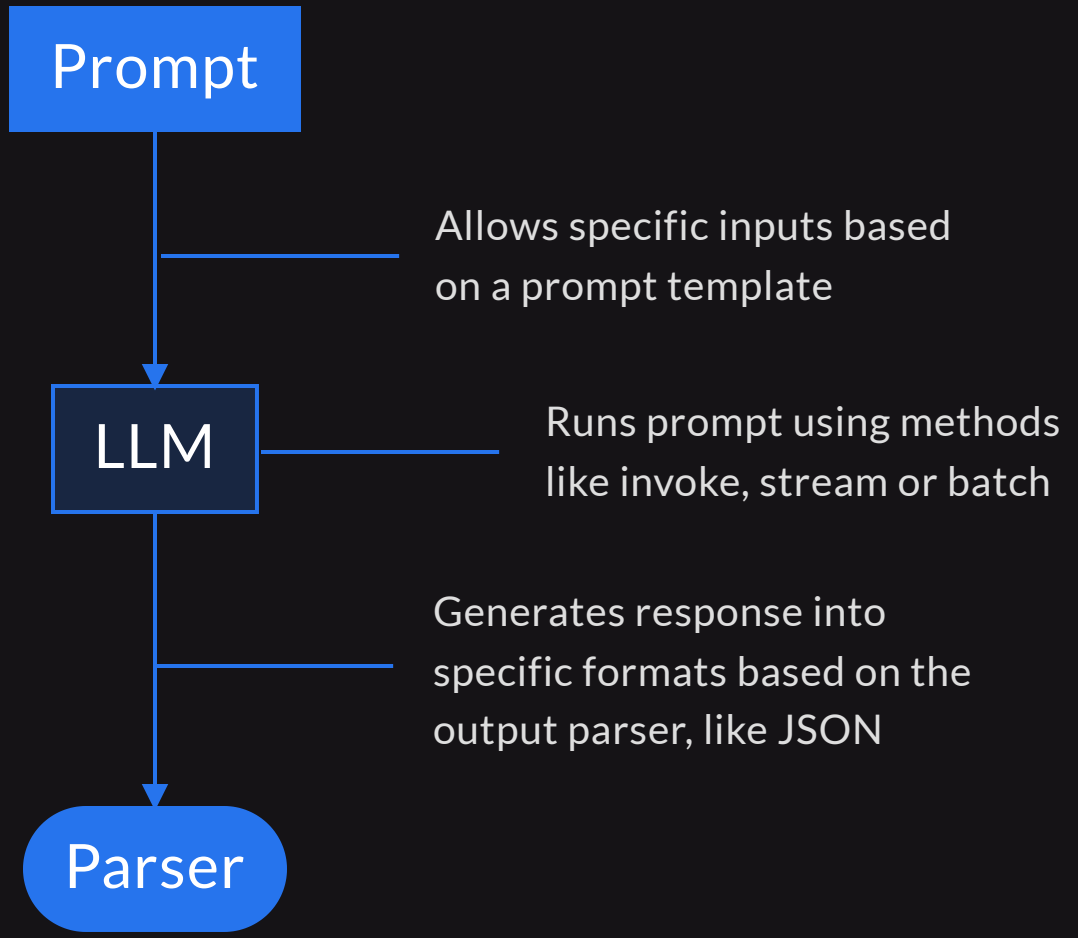
Google Developer Expert - ML & Cloud Champion Innovator

Published Author

Outline

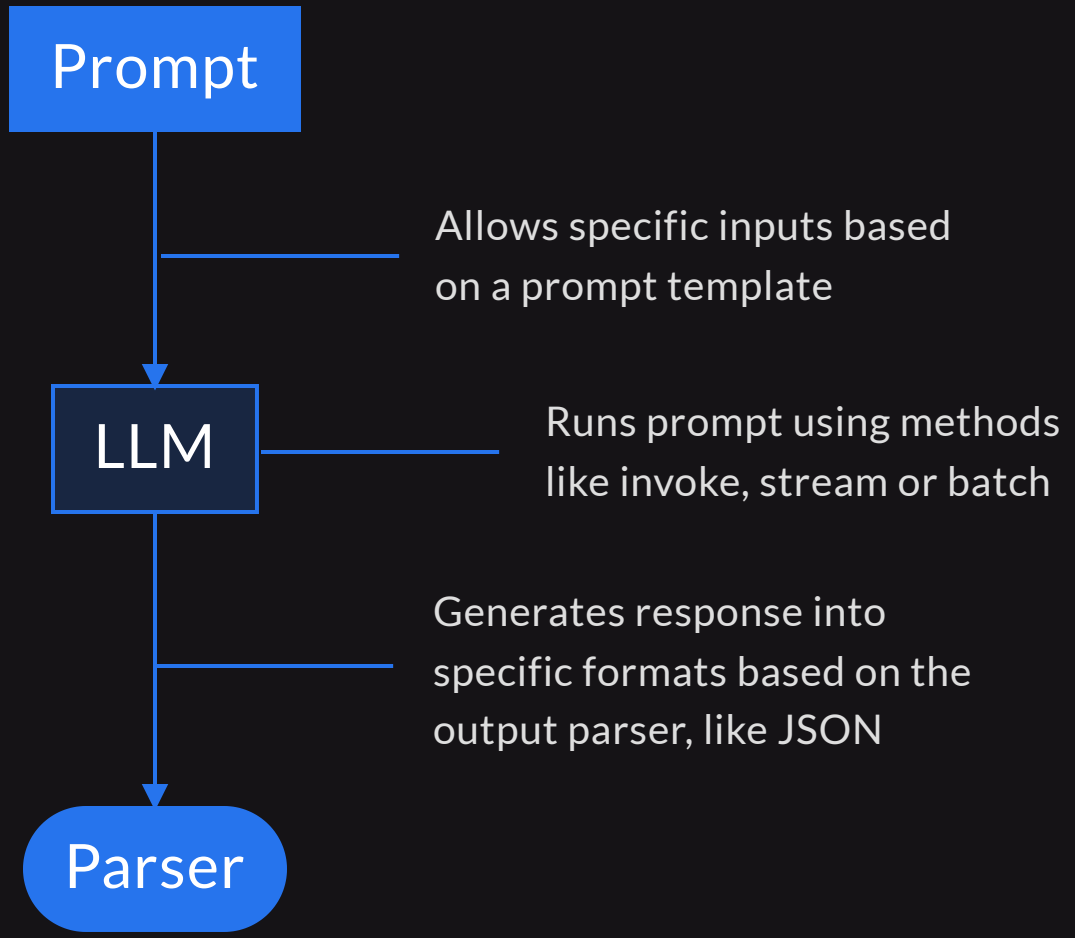
- What is LCEL?
- Typical LCEL Chain Syntax
- LCEL Advantages
- Popular Built-in LCEL Chain Constructors
- LCEL Chain Example

What is LCEL?



- 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

Typical LCEL Chain Syntax



- In LCEL you can chain multiple steps using the overloaded vertical bar or pipe '|' 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

LCEL Advantages

- Streaming support
  - Enable to get chunks of response tokens with the lowest latency and stream the response live to the user
- Async support and parallel execution
  - Support for async APIs to handle many concurrent requests in the same server.
  - You can also execute certain steps in parallel if it is possible
- Deployment, monitoring and observability
  - Can be deployed easily with LangServe.
  - Allow you to access the results of intermediate steps, log them using LangSmith for observability, debugging and monitoring
- Input and Output Schemas
  - Have capabilities to validate inputs and outputs based on specific schemas

Popular Built-in LCEL Chain Constructors

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

LCEL Chain Example

```python
from langchain_openai import ChatOpenAI
from langchain_core.prompts import ChatPromptTemplate

# connect to ChatGPT
chatgpt = ChatOpenAI(model_name="gpt-3.5-turbo", temperature=0)

# create a prompt template to accept user queries
prompt_txt = "{query}"
prompt_template = ChatPromptTemplate.from_template(prompt_txt)

# chain prompt with LLM using LCEL
# you could also write this as llmchain = prompt_template | chatgpt
llmchain = (prompt_template
            |
           chatgpt)

# invoke the chain
response = llmchain.invoke({'query' : 'Explain Generative AI in 1 line'})
print(response.content)
```

Thank You

Analytics Vidhya