Day90_GenAI_LangChain_and_Prompting_Essentials

September 27, 2025

GenAI: LangChain and Prompting Essentials

This notebook covers **LangChain fundamentals** and **prompting techniques**, along with practical steps to build an **LLM-powered financial data extraction app**.
We will go step by step, starting from **prompting basics** to **LangChain setup and usage**.

** Note:**
I would personally use **VS Code** or **Google Colab** for running these notebooks.

- In **VS Code**, I create a `.env` file to store my API keys and load them in the `.py` script. This ensures I can **publicly share the code without exposing API keys**.
- For documentation purposes here, I am using a **Jupyter Notebook** in **Markdown only** and haven't run any cells.
- If you want to try this in **Google Colab**, make sure to **set the runtime to GPU** for faster performance.
- **Never share your API keys publicly**!

Table of Contents (Flow)

- Step 1: Import libraries
- Step 2: Define schema
- Step 3: Create prompt
- Step 4: Create LLM chain
- Step 5: Run with sample input

9. Summary

1 Elements of a Good Prompt

A **prompt** is the input we give to a Large Language Model (LLM).
A good prompt ensures the model understands *context*, *task*, and *format*.

Key elements:

- **Role/Context**: Tell the model *who it is* or *what role it should assume*.
- **Task Instruction**: Be explicit about what you want.
- **Constraints**: Add rules such as word limits, tone, or style.
- **Examples** (if needed): Demonstrate the expected output.
- **Output Format**: JSON, table, bullet points, etc.

Example Prompt:

```
You are a financial analyst.
Summarize the following earnings call transcript into 3 bullet points.
Focus only on revenue, profit, and future outlook.
Return the result in JSON with keys: "revenue", "profit", "outlook".
```

2 Zero-shot, One-shot, and Few-shot Prompting

2.1 Zero-shot Prompting

- No examples given, just instructions.
- Example:

```
Translate "How are you?" into French.
```

2.2 One-shot Prompting

- One example provided to guide the model.

- Example:

```
Translate English to French.

English: Good morning
French: Bonjour

English: How are you?
French:
```

2.3 Few-shot Prompting

- Multiple examples given, helps improve accuracy.
- Example:

```
Translate English to French.

English: Good morning
French: Bonjour

English: I love you
French: Je t'aime

English: How are you?
French:
```

3 LangChain Installation

To install LangChain and dependencies:

```
pip install langchain langchain-community langchain-core
pip install openai groq ollama
```

If you plan to use Jupyter:

```
pip install ipykernel
```

4 Groq and Ollama Setup

4.1 Groq

- Groq provides **fast inference for LLMs**.
- Sign up at Groq, get an API key.
- Set your API key in the environment:

```
export GROQ_API_KEY="your_api_key_here" # never share
```

4.2 Ollama

- Ollama runs **open-source models locally** (like LLaMA, Mistral, etc.).
- Installation (Linux/Mac):

```
curl -fsSL https://ollama.com/install.sh | sh
```

- Run a model:

```
ollama run mistral
```

5 Calling an LLM from LangChain

LangChain provides wrappers to interact with LLMs.

```
from langchain.llms import Ollama

# Initialize model
llm = Ollama(model="mistral")

# Run query
response = llm("Write a short poem about finance.")
print(response)
```

6 Prompt Templates & Chains

Prompt Templates allow you to create reusable prompts with variables.

```
from langchain.prompts import PromptTemplate

template = """You are a financial assistant.
Extract the following information from the text:
Company: {company}
Text: {text}
Return JSON with keys: revenue, profit, outlook.
"""

prompt = PromptTemplate(template=template, input_variables=["company", "text"])

final_prompt = prompt.format(company="Tesla", text="Tesla reported revenue of $25B and profit
print(final_prompt)
```

Chains Chains connect **prompt → LLM → parser → output** into a pipeline.

7 Output Parsers

Output parsers enforce structured results.

```
from langchain.output_parsers import StructuredOutputParser, ResponseSchema

schemas = [
```

```
    ResponseSchema(name="revenue", description="Revenue details"),
    ResponseSchema(name="profit", description="Profit details"),
    ResponseSchema(name="outlook", description="Future outlook"),
]

parser = StructuredOutputParser.from_response_schemas(schemas)
format_instructions = parser.get_format_instructions()

prompt = f"""
Extract financial data.

Text: Tesla reported revenue of $25B and profit of $2B. Outlook is positive.
{format_instructions}
"""

print(prompt)
```

8 Build a Financial Data Extraction App (Step by Step)

We will build a simple **financial data extractor** using LangChain.

Step 1: Import libraries

```
from langchain.llms import Ollama
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from langchain.output_parsers import StructuredOutputParser, ResponseSchema
```

Step 2: Define schema

```
schemas = [
    ResponseSchema(name="revenue", description="Revenue details"),
    ResponseSchema(name="profit", description="Profit details"),
    ResponseSchema(name="outlook", description="Future outlook"),
]

parser = StructuredOutputParser.from_response_schemas(schemas)
format_instructions = parser.get_format_instructions()
```

Step 3: Create prompt

```
template = """You are a financial analyst.
Extract revenue, profit, and outlook from the following text.

Text: {text}

{format_instructions}
"""

prompt = PromptTemplate(
```

```
    template=template,
    input_variables=["text"],
    partial_variables={"format_instructions": format_instructions},
)
```

Step 4: Create LLM chain

```
llm = Ollama(model="mistral")
chain = LLMChain(llm=llm, prompt=prompt)
```

Step 5: Run with sample input

```
text = "Apple reported revenue of $90B and profit of $25B. The outlook is strong with growth i
result = chain.run(text=text)
parsed = parser.parse(result)

print(parsed)
```

Expected Output:

```
{
  "revenue": "$90B",
  "profit": "$25B",
  "outlook": "strong with growth in services"
}
```

9 Summary

- Learned **prompting techniques** (zero, one, few-shot).
- Installed **LangChain, Groq, Ollama**.
- Used **Prompt Templates & Chains**.
- Added **Output Parsers** for structured data.
- Built a **Financial Data Extraction App** using LangChain.