Notes on Prompts, PromptTemplate, and ChatPromptTemplate in LangChain

Comprehensive and Modern Overview

1 Introduction

Prompts form the foundation of modern Large Language Model (LLM) interaction. They serve as carefully crafted instructions or queries that guide a model's reasoning, tone, and output format. In LangChain, prompt engineering is elevated through the `PromptTemplate` and `ChatPromptTemplate` classes, which allow for dynamic and reusable prompt construction.

2 Understanding Prompts

Definition

A **prompt** is the input instruction or query provided to a language model to specify the desired task, output style, or reasoning scope. It acts as a bridge between human intent and model behavior.

2.1 Types of Prompts

- **Text-based Prompts:**
  - Consist solely of textual input or questions.
  - Example: `"Summarize the following paragraph in one sentence."`
- **Multimodal Prompts:**
  - Combine multiple modalities such as text, images, or audio.
  - Example: Providing an image and asking, `"Describe what is happening in this picture."`

2.2 Static vs. Dynamic Prompts

- **Static Prompts:**
  - Fixed content that does not change between executions.

  - Example: `"Translate this sentence into French."`

- **Dynamic Prompts:**
  - Prompts that adapt or change at runtime based on variables or user input.
  - Example: `"Translate the following sentence into {target_language}: {text}"`.

3 PromptTemplate in LangChain

Definition

The **PromptTemplate** class in LangChain provides a structured mechanism for creating reusable, parameterized prompts. Instead of manually concatenating strings, developers define a template with placeholders that are dynamically filled at runtime.

3.1 Key Benefits

- **Reusability:** The same template can be reused with different variables.
- **Flexibility:** Easily adapt prompts for user-driven or automated workflows.
- **Maintainability:** Centralized control of prompt structure; changes propagate automatically.
- **Validation:** Ensures required input variables are provided and correctly formatted before generating the final prompt, reducing runtime errors.

3.2 Usage Example

Example

```
from langchain.prompts import PromptTemplate

template = "Translate the following {language} text to English: {text}"
prompt = PromptTemplate(
    input_variables=["language", "text"],
    template=template
)

formatted = prompt.format(language="French", text="Bonjour le monde")
print(formatted)
# Output:
# Translate the following French text to English: Bonjour le monde
```

4 ChatPromptTemplate

Definition

The **ChatPromptTemplate** class is designed for multi-turn conversational AI. It allows combining multiple message types (system, human, AI) into a single structured template, supporting complex workflows while maintaining context.

4.1 Key Benefits

- **Structured Conversations:** Easily define multi-role interactions (system, human, AI).
- **Dynamic Variables:** Fill placeholders across messages dynamically at runtime.
- **Modular Pipelines:** Integrates seamlessly with LangChain chains for automated LLM workflows.
- **Context Preservation:** Maintains coherent multi-turn dialogues.

4.2 Usage Example

Example

```
from langchain_core.prompts import ChatPromptTemplate
from langchain_google_genai import ChatGoogleGenerativeAI

model = ChatGoogleGenerativeAI(model=”gemini-2.5-flash”)

chat_prompt = ChatPromptTemplate.from_messages([
    (”system”, ”You are a helpful research assistant.”),
    (”human”, ”Summarize the paper titled {paper_title} in simple terms.”)
])

chain = chat_prompt | model
response = chain.invoke({”paper_title”: ”BERT: Pre-training of
Deep Bidirectional Transformers”})
print(response.content)
```

5 Template

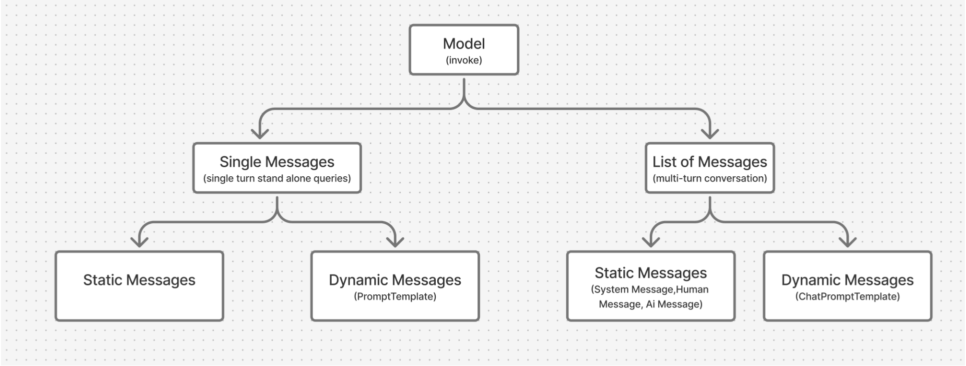

Figure 1: Templates of Prompts

6 Summary

- Prompts define how models interpret and generate responses.
- They can be **static** or **dynamic**, **text-based** or **multimodal**.
- `PromptTemplate` allows structured, reusable prompts.

- `ChatPromptTemplate` enables multi-turn conversational templates with dynamic variables and context management.