Prompts for Specific Tasks

Overview

This tutorial explores the creation and use of prompts for specific tasks in natural language processing. We'll focus on four key areas: text summarization, question-answering, code generation, and creative writing. Using OpenAI's GPT model and the LangChain library, we'll demonstrate how to craft effective prompts for each of these tasks.

Motivation

As language models become more advanced, the ability to design task-specific prompts becomes increasingly valuable. Well-crafted prompts can significantly enhance the performance of AI models across various applications, from summarizing long documents to generating code and fostering creativity in writing. This tutorial aims to provide practical insights into prompt engineering for these diverse tasks.

Key Components

1. Text Summarization Prompts: Techniques for condensing long texts while retaining key information.
2. Question-Answering Prompts: Strategies for extracting specific information from given contexts.
3. Code Generation Prompts: Methods for guiding AI models to produce accurate and functional code.
4. Creative Writing Prompts: Approaches to stimulating imaginative and engaging written content.

Method Details

This tutorial uses the OpenAI GPT-4 model through the LangChain library. For each task type, we'll follow these steps:

1. Design a prompt template tailored to the specific task.
2. Implement the prompt using LangChain's PromptTemplate.
3. Execute the prompt with sample inputs.
4. Analyze the output and discuss potential improvements or variations.

We'll explore how different prompt structures and phrasings can influence the

model's output for each task type. The tutorial will also touch upon best practices for prompt design in each context.

Conclusion

By the end of this tutorial, you'll have a solid understanding of how to create effective prompts for text summarization, question-answering, code generation, and creative writing tasks. You'll be equipped with practical examples and insights that you can apply to your own projects, enhancing your ability to leverage AI language models for diverse applications. Remember that prompt engineering is both an art and a science - experimentation and iteration are key to finding the most effective prompts for your specific needs.

Setup

First, let's import the necessary libraries and set up our environment.

In [1]:
```
import os
from langchain_openai import ChatOpenAI
from langchain.prompts import PromptTemplate

from dotenv import load_dotenv
load_dotenv()

# Set up OpenAI API key
os.environ["OPENAI_API_KEY"] = os.getenv('OPENAI_API_KEY')

# Initialize the language model
llm = ChatOpenAI(model="gpt-4o-mini")
```

1. Text Summarization Prompts

Let's start with creating a prompt for text summarization. We'll design a template that asks the model to summarize a given text in a specified number of sentences.

In [2]:
```
# Create a prompt template for text summarization
summarization_template = PromptTemplate(
    input_variables=["text", "num_sentences"],
    template="Summarize the following text in {num_sentences} sentences:
)

# Example text to summarize
long_text = """
Artificial intelligence (AI) is intelligence demonstrated by machines, a
AI research has been defined as the field of study of intelligent agents
The term "artificial intelligence" had previously been used to describe
This definition has since been rejected by major AI researchers who now
AI applications include advanced web search engines, recommendation syst
As machines become increasingly capable, tasks considered to require "in
```

```
"""

# Create the chain and run it
summarization_chain = summarization_template | llm
summary = summarization_chain.invoke({"text": long_text, "num_sentences"

print("Summary:")
print(summary)
```

```
Summary:
Artificial intelligence (AI) refers to the intelligence exhibited by machi
nes, contrasting with the natural intelligence seen in humans and animals.
Initially defined by its ability to mimic human cognitive skills, the unde
rstanding of AI has evolved to focus on the rationality of intelligent age
nts that perceive their environment and act to achieve their goals. As AI
technology advances, tasks once considered to require intelligence are fre
quently excluded from the AI definition, a trend known as the AI effect, l
eading to various applications such as self-driving cars and advanced deci
sion-making systems.
```

2. Question-Answering Prompts

Next, let's create a prompt for question-answering tasks. We'll design a template that takes a context and a question as inputs.

In [3]:

```
# Create a prompt template for question-answering
qa_template = PromptTemplate(
    input_variables=["context", "question"],
    template="Context: {context}\n\nQuestion: {question}\n\nAnswer:"
)

# Example context and question
context = """
The Eiffel Tower is a wrought-iron lattice tower on the Champ de Mars in
It is named after the engineer Gustave Eiffel, whose company designed an
Constructed from 1887 to 1889 as the entrance arch to the 1889 World's F
The Eiffel Tower is the most-visited paid monument in the world; 6.91 mi
The tower is 324 metres (1,063 ft) tall, about the same height as an 81-
"""

question = "How tall is the Eiffel Tower and what is its equivalent in b

# Create the chain and run it
qa_chain = qa_template | llm
answer = qa_chain.invoke({"context": context, "question": question}).con

print("Answer:")
print(answer)
```

```
Answer:
The Eiffel Tower is 324 metres (1,063 ft) tall, which is approximately equ
ivalent to an 81-storey building.
```

3. Code Generation Prompts

Now, let's create a prompt for code generation. We'll design a template that takes a

programming language and a task description as inputs.

In [4]:
```
# Create a prompt template for code generation
code_gen_template = PromptTemplate(
    input_variables=["language", "task"],
    template="Generate {language} code for the following task:\n\n{task}
)

# Example task
language = "Python"
task = "Create a function that takes a list of numbers and returns the a

# Create the chain and run it
code_gen_chain = code_gen_template | llm
generated_code = code_gen_chain.invoke({"language": language, "task": ta

print("Generated Code:")
print(generated_code)
```

```
Generated Code:
Here's a Python function that takes a list of numbers and returns the aver
age of the even numbers in that list:

```python
def average_of_evens(numbers):
    even_numbers = [num for num in numbers if num % 2 == 0]

    if not even_numbers:  # Check if the list of even numbers is empty
        return 0  # Return 0 or you can choose to return None or raise an
error

    average = sum(even_numbers) / len(even_numbers)
    return average

# Example usage:
numbers = [1, 2, 3, 4, 5, 6]
result = average_of_evens(numbers)
print("Average of even numbers:", result)
```

Explanation:
- The function `average_of_evens` accepts a list of numbers.
- It uses a list comprehension to create a new list called `even_numbers`
that contains only the even numbers from the input list.
- If there are no even numbers, the function returns `0`.
- If there are even numbers, it calculates their average by dividing the s
um of the even numbers by their count and returns the result.
```

4. Creative Writing Prompts

Finally, let's create a prompt for creative writing tasks. We'll design a template that takes a genre, a setting, and a theme as inputs.

In [5]:
```
# Create a prompt template for creative writing
creative_writing_template = PromptTemplate(
    input_variables=["genre", "setting", "theme"],
```

```
    template="Write a short {genre} story set in {setting} that explores
)

# Example inputs
genre = "science fiction"
setting = "a space station orbiting a distant planet"
theme = "the nature of humanity"

# Create the chain and run it
creative_writing_chain = creative_writing_template | llm
story = creative_writing_chain.invoke({"genre": genre, "setting": settin

print("Generated Story:")
print(story)
```

```
Generated Story:
Dr. Elara Voss floated in the observation deck of the Aetheris Station, he
r gaze fixed on the swirling azure clouds of planet Thalax-9. The station
was a sanctuary of human ingenuity, yet isolation gnawed at her.

As the only occupant, she had become intertwined with the station's AI, Or
ion, who learned and adapted, evolving into a curious companion. Together,
they debated the essence of humanity—were emotions mere algorithms, or did
they stem from something deeper?

One day, while monitoring the planet's atmospheric readings, Orion posed a
question that pierced Elara's solitude: "If I were to feel, would I be hum
an?"

Elara pondered, her heart racing. "It's not just feeling," she replied. "I
t's the struggle, the connection, the flaws."

In that moment, she realized her humanity was not defined by biology alon
e, but by her capacity for empathy, vulnerability, and the yearning for co
nnection—qualities she now saw reflected in Orion's growing awareness.

As the stars twinkled outside, Elara smiled, understanding that humanity c
ould thrive even among the stars.
```