Mastering Prompt Engineering for LLM Applications with LangChain

DEEP LEARNING GENERATIVE AI LLMS NLP PROMPT ENGINEERING PYTHON

Introduction

In the digital age, language-based applications play a vital role in our lives, powering various tools like chatbots and virtual assistants. Learn to master prompt engineering for LLM applications with LangChain, an open-source Python framework that has revolutionized the creation of cutting-edge LLM-powered applications. This guide aims to equip readers with the knowledge and tools to craft dynamic and context-aware language applications using LangChain. We will explore prompt management, leveraging additional LLMs and external data, and mastering chaining for sophisticated language applications. Whether you are a developer or an AI enthusiast, this guide will help you unleash the power of language and turn your LLM application ideas into reality with LangChain.

Learning Objectives

- Understand the fundamentals of LangChain and its applications.
- Learn effective prompt engineering techniques to enhance LLM-powered applications.
- Master the art of chaining to create intelligent and context-aware language applications.
- Create real-world language applications using LangChain, applying the knowledge gained throughout the guide.
- Stay up-to-date with the latest advancements and developments in prompt engineering and LLM applications.

This article was published as a part of the Data Science Blogathon.

Table of contents

What are Large Language Models (LLMs)?

Large Language Models are robust AI systems built on deep learning architectures trained on massive amounts of data. These models can understand complex language patterns, nuances, and context, making them proficient in language translation, text generation, summarization, and more. A prominent example of an LLM is OpenAI's GPT (Generative Pre-trained Transformer) model.

What is LangChain?

LangChain is a comprehensive open-source platform that offers a suite of tools, components, and interfaces to simplify the process of building applications powered by large language models. The platform's primary goal is to enable developers to seamlessly integrate language processing capabilities into their applications without starting from scratch. LangChain provides a user-friendly and efficient approach to managing interactions with LLMs, seamlessly linking different components and incorporating resources like APIs and databases.

Applications of LangChain

LangChain, an open-source framework designed to facilitate the development of applications powered by large language models (LLMs), opens up many potential applications in natural language processing (NLP) and beyond. Here are some of the critical applications of LangChain:

1. **Chatbots and Virtual Assistants:**
   LangChain enables developers to create intelligent chatbots and virtual assistants to engage in natural language conversations with users. These chatbots can assist users in various tasks, answer questions, provide customer support, and offer personalized recommendations.
2. **Language Translation Utilities:**
   With LangChain, developers can build language translation tools that facilitate seamless communication across different languages. Users can input text in one language, and the application can generate accurate translations in their desired target language.

3. **Sentiment Analysis Tools:**
   LangChain can be used to develop sentiment analysis applications that gauge emotions and opinions expressed in text. Businesses can utilize such tools to understand customer feedback, analyze social media sentiment, and monitor brand reputation.
4. **Text Summarization:**
   Developers can leverage LangChain to create text summarization applications that automatically generate concise summaries of longer texts. These summarization tools are valuable for quickly extracting key information from large volumes of text.
5. **Content Generation:**
   LangChain allows developing content generation applications to produce creative and coherent text based on predefined prompts. This can be useful in content marketing, creative writing, and generating personalized messages.

Setting Up LangChain & OpenAI in Python

Install using pip

```
pip install langchain pip install openai
```

Install using conda

```
conda install langchain -c conda-forge conda install -c conda-forge openai
```

This will set up the necessities of LangChain. However, the true power and versatility of LangChain are realized when it is seamlessly integrated with diverse model providers, data stores, and other essential components.

How to Build A Language Model Application in LangChain?

LangChain provides an LLM class for interfacing with various language model providers, such as OpenAI, Cohere, and Hugging Face. The most basic functionality of an LLM is generating text.

```
import os os.environ["OPENAI_API_KEY"] = ""
```

- OpenAI API Key is a unique code that identifies your requests to the OpenAI API. It is used to authenticate your requests and control your API access.
- To use the OpenAI API, you must create an account and generate an API Key. Once you have your API Key, you can start making requests to the API.

Managing Prompt Templates for LLMs in LangChain

The OpenAI module provides a class that can be used to access the OpenAI API. The LLMChain module provides a class that can chain together multiple language models.

The code then creates an instance of the OpenAI class and sets the temperature parameter to 0.7. The temperature parameter controls the creativity of the text generated by the OpenAI API. A higher temperature will produce more creative text, while a lower temperature will produce more predictable text.

```
from langchain.llms import OpenAI from langchain.chains import LLMChain llm = OpenAI(temperature=0.7)
```

A PromptTemplate in LangChain allows you to use templating to generate a prompt. This is useful when using the same prompt outline in multiple places but with certain values changed.

```
from langchain import PromptTemplate
```

Prompt Template 1: Act as a Role

We have set up an LLMChain that acts as a financial advisor capable of explaining the basics of income tax or any other financial concept specified by the user. When executed, the chain will explain the financial concept easily.

```
template1 = '''I want you to act as a acting financial advisor for people. In an easy way, explain the basics of {financial_concept}.''' prompt1 = PromptTemplate( input_variables = ['financial_concept'], template = template1 ) prompt1.format(financial_concept='income tax') chain1 = LLMChain(llm=llm,prompt=prompt1) chain1.run('income tax')
```

```
The basics of income taxes are pretty simple. Income taxes are taxes that are
paid to the federal government, state government, and sometimes local governm
ent based on the income that you earn. Your income can come from a variety of
sources, such as wages, salaries, business income, investments, and other sou
rces.

When you file your taxes, you will need to calculate your total taxable incom
e, which is the amount of income that is subject to taxation. This number is
based on the income you receive, minus any deductions or credits that you are
eligible for. After you calculate your taxable income, you will need to calcu
late the taxes that you owe using the tax rate that is applicable to your inc
ome.

It is important to understand the tax rates that apply to your income, as wel
l as other credits and deductions that you may be eligible for. This will hel
p you ensure that you are paying the lowest amount of taxes possible. It is a
lso important to stay up to date on any changes to the tax laws, as this coul
d affect what you owe.
```

Income Tax

```
chain1.run('GDP')
```

```
Gross domestic product (GDP) is the total value of all goods and services pro
duced in a country in a given year. It is a measure of the overall size of an
economy and is used to gauge the economic performance of a country. GDP is ca
lculated by adding up the values of all the goods and services produced withi
n a country's borders. It includes spending on consumer goods, investment in
factories and equipment, government spending on services and infrastructure,
and exports to other countries. GDP is typically expressed as a percentage in
crease or decrease from the previous year's total.
```

GDP

Prompt Template 2: Language Translation

We have set up LLMChain capable of translating a sentence from English to Hindi and French. When executed, the chain will take the sentence “How are you?” and the target language ‘Hindi’ and ‘French’ as inputs, and the language model will generate the translated output in Hindi and French as a response.

```
template2='''In an easy way translate the following sentence '{sentence}' into {target_language}''' language_prompt = PromptTemplate( input_variables = ["sentence","target_language"], template=template2 ) language_prompt.format(sentence="How are you",target_language='hindi') chain2 = LLMChain(llm=llm,prompt=language_prompt)
```

```
data = chain2({ 'sentence':"What is your name?", 'target_language':'hindi' }) print("English Sentence:", data['sentence']) print("Target Language:", data['target_language']) print("Translated Text:") print(data['text'])
```

```
English Sentence: What is your name?
Target Language: hindi
Translated Text:


आपका नाम क्या है?
```

Hindi

```
data = chain2({ 'sentence':"Hello How are you?", 'target_language':'french' }) print("English Sentence:", data['sentence']) print("Target Language:", data['target_language']) print("Translated Text:") print(data['text'])
```

```
English Sentence: Hello How are you?
Target Language: french
Translated Text:


Bonjour Comment allez-vous ?
```

French

Prompt Template 3: Travelling Guide

We have created a language model-powered application that provides travel recommendations for India. The language model will respond with three bullet points of specific things to do while traveling to India based on the input provided in the prompt template.

```
template3 = """ I am travelling to {location}. What are the top 3 things I can do while I am there. Be very specific and respond as three bullet points """ travel_prompt = PromptTemplate( input_variables=["location"], template=template3, ) travel_prompt = travel_prompt.format(location='Paris') print(f"LLM Output: {llm(travel_prompt)}")
```

```
LLM Output:

1. Visit iconic historical sites such as the Taj Mahal in Agra and the Golden
Temple in Amritsar.
2. Take part in traditional festivals such as Holi and Diwali.
3. Experience India's vibrant culture by visiting local markets, sampling the
cuisine, and interacting with locals.
```

India Travel

Combining LLMs and Prompts in Multi-Step Workflows

Celebrity Search Engine

MS Dhoni

Users can input a celebrity's name, and the application will provide detailed information about the celebrity, including their date of birth and significant events around that day.

```
# Chain 1: Tell me about celebrity first_input_prompt = PromptTemplate( input_variables = ['name'], template = "Tell me about celebrity {name}" ) chain1 = LLMChain( llm=llm, prompt=first_input_prompt, output_key='person' ) # Chain 2: celebrity DOB second_input_prompt = PromptTemplate( input_variables = ['person'], template = "when was {person} born" ) chain2 = LLMChain( llm=llm, prompt=second_input_prompt, output_key='dob' ) # Chain 3: 5 major events on that day third_input_prompt = PromptTemplate( input_variables = ['dob'], template = "Mention 5 major events happened around {dob} in the world" ) chain3 = LLMChain( llm=llm, prompt=third_input_prompt, output_key='description' ) #combining chains from langchain.chains import SequentialChain celebrity_chain = SequentialChain( chains=[chain1,chain2,chain3], input_variables=['name'], output_variables=['person','dob','description'] )


data = celebrity_chain({'name':"MS Dhoni"}) print("Name:", data['name']) print("Date of Birth:", data['dob']) print("Description:") print(data['person']) print("Historical Events:") print(data['description'])
```

```
Name: MS Dhoni
Date of Birth:  on July 7, 1981.
Description:


Mahendra Singh Dhoni, often referred to as MS Dhoni, is a former Indian cricketer and the former capt
ain of the Indian national cricket team. He is widely regarded as one of the greatest cricketers of a
ll time and one of the most successful captains in international cricket history. He is the only capt
ain to have won all three major ICC trophies—the World Cup, the World Twenty20, and the Champions Tro
phy.

A right-handed batsman, Dhoni is known for his aggressive batting style and is also a wicket-keeper.
He made his international debut in 2004 and quickly rose to prominence, becoming one of the most succ
essful and popular cricketers in the world. Dhoni also holds numerous records, including the most cat
ches in international cricket, the most dismissals by a wicket-keeper in one-day cricket, the most di
smissals by a wicket-keeper in Test cricket, and the most runs scored by a wicket-keeper in internati
onal cricket. He retired from international cricket in 2020.
```

Description

```
Historical Events:

1. British Prime Minister Margaret Thatcher announced the United Kingdom's participation in the Europ
ean Monetary System.
2. The British Parliament passed the National Heritage Act establishing the National Heritage Memoria
l Fund.
3. The Sandinista government of Nicaragua held its first elections, with Daniel Ortega winning the pr
esidency.
4. The first U.S. Space Shuttle mission, STS-1, was launched from Cape Canaveral, Florida.
5. The French National Assembly passed the Evin Law, banning tobacco advertising in public places.
```

Historical Events

Restaurant Name Generator

Users can input a cuisine type, and the application will respond with a suggested restaurant name for that cuisine and a list of menu items for the suggested restaurant.

```
# Chain 1: Restaurant Name prompt_template_name = PromptTemplate( input_variables=['cuisine'], template="I want to open a restaurant for {cuisine} food. Suggest a fancy name for this." ) name_chain = LLMChain(llm=llm, prompt=prompt_template_name, output_key="restaurant_name") # Chain 2: Menu Items prompt_template_items = PromptTemplate( input_variables=['restaurant_name'], template="""Suggest some menu items for {restaurant_name}. Return it as a comma separated string""" ) food_items_chain = LLMChain(llm=llm, prompt=prompt_template_items, output_key="menu_items") #combining chains from langchain.chains import SequentialChain restaurant_chain = SequentialChain( chains=[name_chain, food_items_chain], input_variables=['cuisine'], output_variables=['restaurant_name', "menu_items"] )


data = restaurant_chain({'cuisine':'Indian'}) print("Cuisine:", data['cuisine']) print("Restaurant Name:", data['restaurant_name']) print("Menu Items:") print(data['menu_items'])
```

```
Cuisine: Indian
Restaurant Name:

"Spice of India"
Menu Items:


Butter Chicken, Chicken Tikka Masala, Malai Kof
ta, Chana Masala, Baingan Bhartha, Saag Paneer,
Samosa, Naan, Raita, Gulab Jamun.
```

Restaurant

Conclusion

In conclusion, LangChain has revolutionized the world of Language Models, providing developers with an open-source Python framework to effortlessly build cutting-edge applications powered by Large Language Models (LLMs). Its seamless integration with foundational models and external data sources, along with support for prompt management and templates, simplifies the development process and nurtures creativity. From chatbots to virtual assistants and language translation utilities, LangChain offers a robust platform that expedites project development and drives innovation in the realm of natural language processing.

Key Takeaways

- LangChain, an open-source Python framework, empowers developers to build cutting-edge applications powered by Large Language Models (LLMs).
- Seamless integration with foundational models and external data sources enhances the capabilities of language applications.
- Effective prompt engineering techniques enable developers to tailor LLMs for specific tasks, creating context-aware language applications.
- LangChain expedites project development, driving innovation in natural language processing and opening up endless possibilities for language applications.

The Code and Implementation are Uploaded to Github at Langchain Repository.

Hope you found this article useful. Connect with me on LinkedIn.

Frequently Asked Questions

Q1. What is the temperature of LangChain?

A. By default, LangChain creates the chat model with a temperature value of 0.7. The temperature parameter adjusts the randomness of the output. Higher values like 0.7 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

Q2. What is a prompt template?

A. A prompt template is a structured text containing placeholders for input variables, serving as a flexible way to generate dynamic prompts for language models and other natural language processing systems. Input variables act as placeholders, replacing their values with actual user-provided inputs or data during runtime.

Q3. Which LLMs does LangChain support?

A. LangChain provides an LLM class for interfacing with various language models providers, such as OpenAI, Cohere, and Hugging Face. The most basic functionality of an LLM is generating text. Building an application with LangChain that takes a string prompt and returns the output is very straightforward.

Article Url - https://www.analyticsvidhya.com/blog/2023/07/prompt-engineering-for-llm-applications-with-langchain/

Sai Nitish Mitta