



7.14






 基本LongChain的文档系统.md




 文本内容如何转化为语义向量.md



 文档系统数据输入输出对比.md

大型语言模型（LLM）理论简介

 00.大型语言模型（LLM）理论简介.pptx

Model IO -datastream

0.IPO -langchain

1. IPO

I: input==提示词 提示模板 + 调用时传入 占位符 的实际值

P: process==model / chain (workflow+agent)

Model 以dirct /chain / agent 方式提供和处理

调用方法

O: output ==调用/执行的结果

数据结构（提取model相应的文本，本次的输出可能作为后续的输入）

2. 步骤：

1.创建提示

2.创建模型

3.模型调用提示

3. Prompt

1.分类： prompt ---单轮问答

chatprompt ---多轮问答（角色：系统消息，用户消息，AI消息） prompt的子类

2.四个构件元素：

模板字符串：template_str ,带有占位符的字符串，供提示词模板类使用

提示词模板类：PromptTemplate,借助/基于 模板字符串 生成 提示词模板（实例）对象

提示词模板（实例）对象：prompt_template[late],用来生成最终/最后的提示词字符串（真实 实际 提示词）

提示词字符串：prompt_str,真实的、占位符被实际值填充的实际使用的提示词

3.创建提示的流程/步骤

（1）依据PromptTemplate中的带有占位符的template_str创建prompt_temp

依据模板字符串，借助提示词模板类，创建提示词模板对象

（2）调用提示词模板对象的格式化方法，填入占位符

生成提示词字符串

task01:直接调用model,提问：课程内容 ==PromptTemplate

```

1  # 0.导入环境信息
2  import os
3  from dotenv import load_dotenv
4
5  # Load environment variables from .env file
6  load_dotenv()
7
8  #1.生成/创建提示
9  #1-1 生成提示词模板字符串
10 template_str: str="{course}课程的内容是? "
11 #1-2 生成提示词模板（实例）对象
12 from langchain_core.prompts import PromptTemplate
13 prompt_template :PromptTemplate= PromptTemplate.from_template(
14     template=template_str
15 )
16 #1-3 生成提示词字符串 --最终/实际使用的提示词
17 #参数：对应 模板字符串中的占位符的名称
18 course_name:str="数据结构"
19 prompt_str:str=prompt_template.format(
20     course=course_name #写占位符的名字
21 )
22 #打印输出 提示词的字符串
23 print(prompt_str)
24
25 #2.生成/装载模型
26 from langchain_openai import ChatOpenAI
27 chat_model=ChatOpenAI(
28     model="qwen-plus",
29     base_url=os.environ["OPENAI_BASE_URL_FREE"],
30     api_key=os.environ["OPENAI_APIKEY_FREE"],
31     temperature=0
32 )
33
34 #3.传入提示,调用模型,返回响应
35 from langchain_core.messages import AIMessage
36 response:AIMessage=chat_model.invoke(
37     input=prompt_str
38 )
39
40 print(response)
41 result:str=response.content
42 print(result)

```

task02:直接调用model,提问：课程内容 ==ChatPromptTemplate

系统消息，用户消息，AI消息 ==提示词模板（类，对象），提示词字符串

需要将上述三种角色的提示封装到chatprompt

le09_chatprompt_dirctmodel

```
1  # 0. 导入环境信息
2  import os
3  from dotenv import load_dotenv
4  from langchain_core.messages import SystemMessage
5
6  # Load environment variables from .env file
7  load_dotenv()
8  from langchain_core.prompts import
    SystemMessagePromptTemplate, HumanMessagePromptTemplate, ChatPromptTemplate, AIMessagePromptTemplate
9  #1. 生成提示
10 #1-1 生成 系统角色 提示
11 #1-1-1 生成 系统角色 提示 模板字符串
12 system_message_template_str= "你是一个计算机专业教师，擅长回答计算机专业课程的问题。"
13 #1-1-2 生成 系统角色 提示 模板对象
14 system_message_prompt_template =
    SystemMessagePromptTemplate.from_template(template=system_message_template_str)
15
16 #1-2 生成 用户角色 提示
17 #1-2-1 生成 用户角色 提示 模板字符串
18 human_message_template_str= "{course}课程核心知识点是？"
19 #1-2-2 生成 用户角色 提示 模板对象
20 human_message_prompt_template =
    HumanMessagePromptTemplate.from_template(template=human_message_template_str)
21
22 #1-3 封装组合 chat 的模板 对象
23 chat_prompt_template=ChatPromptTemplate.from_messages(
24     messages=[system_message_prompt_template,
25               human_message_prompt_template,
26             ]
27 )
28
29 # 或者
30 # chat_prompt_template= chat_prompt_template.from_messages(
31 #     messages=[
32 #         ("system", "你是一个计算机专业教师，擅长回答计算机专业课程的问题。"),
33 #         ("human", "{course}课程核心知识点是？")
34 #     ]
35 # )
36
37 #1-4 生成提示字符串
38 chat_prompt_str=chat_prompt_template.format_messages(
39     course="数据结构"
```

```
40 )
41
42 #2.生成/装载模型
43 from langchain_openai import ChatOpenAI
44 chat_model=ChatOpenAI(
45     model="qwen-plus",
46     base_url=os.environ["OPENAI_BASE_URL_FREE"],
47     api_key=os.environ["OPENAI_APIKEY_FREE"],
48     temperature=0
49 )
50
51 #3.传入提示,调用模型,返回响应
52 from langchain_core.messages import AIMessage
53 response:AIMessage=chat_model.invoke(
54     input=chat_prompt_str
55 )
56
57 print(response)
58 result:str=response.content
59 print(result)
```

thinking:

Langchain app 输入的起点?

输入的内容怎么看? 输出是啥? 应该怎么看?