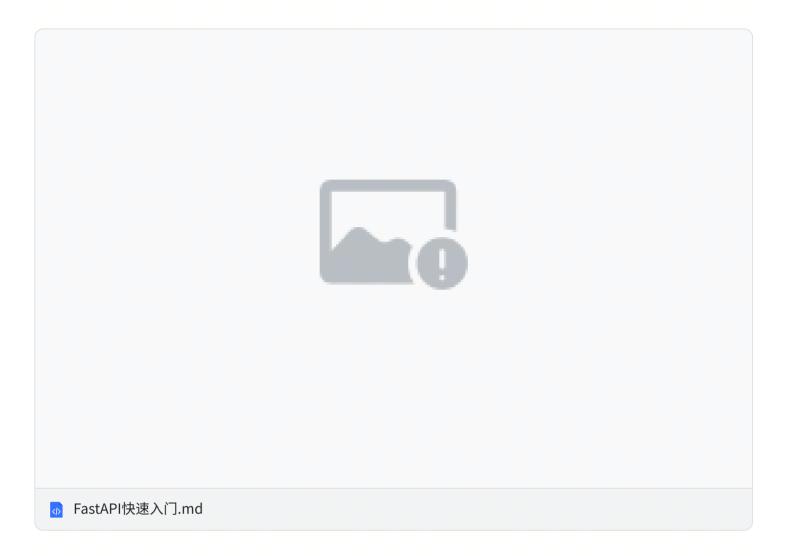
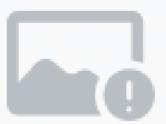
## 7.14





基本LongChain的文档系统.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

P 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dircct /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,Al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: PromptTemlptae,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

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

#### 需要将上述三种角色的提示封装到chatprompt

```
le09 chatprompt dirctmodel
    # 0.导入环境信息
1
    import os
2
3
    from dotenv import load_dotenv
    from langchain core.messages import SystemMessage
 4
 5
    # Load environment variables from .env file
 6
7
    load_dotenv()
    from langchain_core.prompts import
 8
    SystemMessagePromptTemplate, HumanMessagePromptTemplate, ChatPromptTemplate, AIMes
    sagePromptTemplate
    #1.生成提示
9
    #1-1 生成 系统角色 提示
10
    #1-1-1 生成 系统角色 提示 模板字符串
11
    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。"
12
    #1-1-2 生成 系统角色 提示 模板对象
13
    system_message_prompt_template =
14
    SystemMessagePromptTemplate.from_template(template=system_message_template_str)
15
    #1-2 生成 用户角色 提示
16
17
    #1-2-1 生成 用户角色 提示 模板字符串
    human_message_template_str= "{course}课程核心知识点是? "
18
    #1-2-2 生成 用户角色 提示 模板对象
19
    human_message_prompt_template =
20
    HumanMessagePromptTemplate.from_template(template=human_message_template_str)
21
    #1-3 封装组合 chat 的模板 对象
22
    chat_prompt_template=ChatPromptTemplate.from_messages(
23
        messages=[system_message_prompt_template,
24
25
                 human_message_prompt_template,
26
        ٦
    )
27
28
    # 或者
29
    # chat_prompt_template= chat_prompt_template.from_messages(
30
31
         messages=[
              ("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。"),
32
              ("human", "{course}课程核心知识点是?")
33
          7
34
    #
    # )
35
36
37
    #1-4 生成提示字符串
38
    chat_prompt_str=chat_prompt_template.format_messages(
        course="数据结构"
39
```

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