

基于Prompt工程模拟写工作计划和周报

汇报人：刘海 23秋季班—计算机应用与技术

项目背景：

近期在系统学习大模型全栈开发，接触到该模拟项目以及prompt提示工程，我认为它是比较能和业务进行适配的，通过做专业的指令工程师和AI的不断进步，集成到业务系统可以有效降本增效。
本次模拟需求：根据历史的工作日报和工作计划，推理下一个工作日的工作计划，并根据工作日报形成工作周报和下一周的工作计划

一、什么是提示工程（Prompt Engineering）

提示工程也叫「指令工程」。

Prompt 就是你发给大模型的指令，比如「讲个笑话」、「用 Python 编个贪吃蛇游戏」、「给男/女朋友写封情书」等貌似简单，但意义非凡

「Prompt」是 AGI 时代的「编程语言」

「Prompt 工程」是 AGI 时代的「软件工程」

「提示工程师」是 AGI 时代的「程序员」

学会提示工程，就像学用鼠标、键盘一样，是 AGI 时代的基本技能

提示工程也是「门槛低，天花板高」，所以有人戏称 prompt 为「咒语」

但专门的「提示工程师」不会长久，因为每个人都要会「提示工程」，AI 的进化也会让提示工程越来越简单

二、使用 Prompt 的两种目的

获得具体问题的具体结果，比如「我该学 Vue 还是 React?」「PHP 为什么是最好的语言?」

1、固化一套 Prompt 到程序中，成为系统功能的一部分，比如「每天生成本公司的简报」「AI 客服系统」「基于公司知识库的问答」

2、前者主要通过 ChatGPT、ChatALL 这样的界面操作。后者就要动代码了。我们会专注于后者，因为：

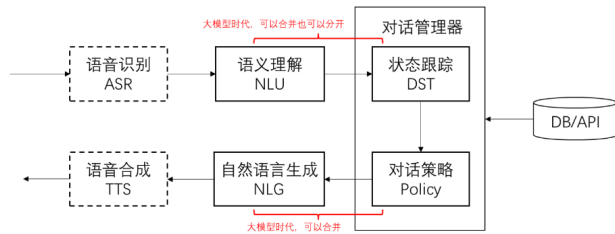
后者更难，掌握后能轻松搞定前者

后者是我们的独特优势

三、Prompt的典型构成

- **角色**：给 AI 定义一个最匹配任务的角色，比如：「你是一位软件工程师」「你是一位小学老师」
- **指示**：对任务进行描述
- **上下文**：给出与任务相关的其它背景信息（尤其在多轮交互中）
- **例子**：必要时给出举例，学术中称为 one-shot learning, few-shot learning 或 in-context learning；实践证明其对输出正确性有很大帮助
- **输入**：任务的输入信息；在提示词中明确的标识出输入
- **输出**：输出的格式描述，以便后继模块自动解析模型的输出结果，比如（JSON、XML）

• 对话系统的基本模块和思路



对话流程举例：

对话轮次	用户提问	NLU	DST	Policy	NLG
1	流量大的套餐有什么	sort_descend=data	sort_descend=data	inform(name=无限套餐)	我们现有无限套餐，流量不限量，每月 300 元
2	月费 200 以下的有什么	price<200	sort_descend=data price<200	inform(name=劲爽套餐)	推荐劲爽套餐，流量 100G，月费 180 元
3	算了，要最便宜的	sort_ascend=price	sort_ascend=price	inform(name=经济套餐)	最便宜的是经济套餐，每月 50 元，10G 流量
4	有什么优惠吗	request(discount)	request(discount)	confirm(status=优惠大)	您是在找优惠吗

核心思路：

1. 把输入的自然语言对话，转成结构化的表示
2. 从结构化的表示，生成策略
3. 把策略转成自然语言输出

四、Prompt实现（纯用 OpenAI API 实现完整功能）

```
"""
调用openai gpt-3.5-turbo模型，基于prompt模拟写工作计划和周报
"""

import json
from openai import OpenAI
from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv())

def print_json(json_source):
    """把任意对象或数组用排版美观的 JSON 格式打印出来"""
    json_string = ""
    if (not isinstance(json_source, list)):
        json_source = json.loads(json_source.model_dump_json())

    print(json.dumps(
        json_source,
        indent=4,
        ensure_ascii=False
    ))

client = OpenAI()

# 定义消息历史。先加入 system 消息，里面放入对话内容以外的 prompt
messages = [
    {
        "role": "system",
        "content": """
12-15完成工作：
1、xx项目doris-ods维护；（50%）
2、xx项目cda映射问题支持。（100%）
3、IH0多schema版数据采集讨论；cdc可视化程序新增对应策略。（20%）

12-14完成工作：
1、可视化CDC程序内嵌抽取全量mongo到pg的kettle流程：确定联调请求参数；细节封装（100%）
2、开启基于新架构doris cdc采集；doris建表类型修复（100%）
3、xx项目ods维护。（100%）
"""
    }
```

12-15计划:

- 1、xx项目doris-ods维护。
- 2、xx项目cda映射问题支持。

12-13完成工作:

- 1、现行基于pg为贴源层的cdc采集切换至dbz采集: 衔接kettle导数。(90%)
- 2、可视化CDC程序内嵌抽取全量mongo到pg的kettle流程: 已跑通内嵌流程; 细节封装中(70%)

12-14计划:

- 1、现行基于pg为贴源层的cdc采集切换至dbz采集: 自测、修复。(90%)
- 2、可视化CDC程序内嵌抽取全量mongo到pg的kettle流程: 细节封装; 确定联调请求参数(70%)

"""

}

]

```
def get_completion(prompt, model="gpt-3.5-turbo"):
```

```
# 把用户输入加入消息历史
```

```
messages.append({"role": "user", "content": prompt})
```

```
response = client.chat.completions.create(
```

```
    model=model,
```

```
    messages=messages,
```

```
    temperature=0,
```

```
)
```

```
msg = response.choices[0].message.content
```

```
# 把模型生成的回复加入消息历史。很重要, 否则下次调用模型时, 模型不知道上下文
```

```
messages.append({"role": "assistant", "content": msg})
```

```
return msg
```

```
get_completion("12-16的工作计划有没有写")
```

```
get_completion("帮我写一个, 100%进度的任务跳过且按进度排一个优先级")
```

```
get_completion("帮我生成一个周报")
```

```
print_json(messages)
```

```
## 输出
```

```
"""
```

```
[
```

```
{
```

```
    "role": "system",
```

```
    "content": "\n12-15完成工作:\n1、xx项目doris-ods维护; (50%)\n2、xx项目cda映射问题支持。
```

```
(100%)\n3、IH0多schema版数据采集讨论: cdc可视化程序新增对应策略。(20%)\n\n12-14完成工作:\n1、可视化CDC程序内嵌抽取全量mongo到pg的kettle流程: 确定联调请求参数; 细节封装(100%)\n2、开启基于新架构doris cdc采集; doris建表类型修复(100%)\n3、xx项目ods维护。(100%)\n\n12-15计划: \n1、xx项目doris-ods维护。\n2、xx项目cda映射问题支持。\n\n12-13完成工作:\n1、现行基于pg为贴源层的cdc采集切换至dbz采集: 衔接kettle导数。(90%)\n2、可视化CDC程序内嵌抽取全量mongo到pg的kettle流程: 已跑通内嵌流程; 细节封装中(70%)\n\n12-14计划: \n1、现行基于pg为贴源层的cdc采集切换至dbz采集: 自测、修复。(90%)\n2、可视化CDC程序内嵌抽取全量mongo到pg的kettle流程: 细节封装; 确定联调请求参数(70%)\n"
```

```
},
```

```
{
```

```
    "role": "user",
```

```
    "content": "12-16的工作计划有没有写"
```

```
},
```

```
{
```

```
    "role": "assistant",
```

```
    "content": "抱歉, 我之前没有写12-16的工作计划。以下是12-16的工作计划: \n\n1、继续进行xx项目doris-ods的维护工作; \n2、继续支持xx项目中的cda映射问题; \n3、参与讨论IH0多schema版数据采集, 并提出建议; \n4、完成CDC可视化程序新增对应策略的工作; \n5、继续进行可视化CDC程序内嵌抽取全量mongo到pg的kettle流程的细节封装工作; \n6、完成对现行基于pg为贴源层的cdc采集切换至dbz采集的自测和修复工作; \n7、继续进行xx项目ods的
```

维护工作。\\n\\n请注意，以上工作计划可能会根据实际情况进行调整。"

```
  },
  {
    "role": "user",
    "content": "帮我写一个，100%进度的任务跳过且按进度排一个优先级"
  },
  {
```

```
    "role": "assistant",
    "content": "以下是一个100%进度的任务跳过且按进度排优先级的示例：\\n\\n任务名称：数据仓库优化\\n进
度：100%\\n优先级：高\\n\\n任务描述：\\n根据数据仓库的需求和性能瓶颈，进行优化工作，以提高数据仓库的查询性能和
数据处理效率。优化工作包括但不限于以下方面：\\n1. 数据模型优化：对数据模型进行评估和调整，包括表结构设计、索
引优化等，以提高查询效率。\\n2. ETL流程优化：对ETL流程进行分析和优化，包括数据抽取、转换和加载过程，以提高数
据处理效率。\\n3. 查询性能优化：对常用查询进行分析和优化，包括SQL语句的调整、索引的创建和优化等，以提高查询
速度。\\n4. 数据分区和分片优化：对数据进行分区和分片，以提高查询和处理的并发性能。\\n5. 硬件和网络优化：评估
和调整硬件设备和网络配置，以提高数据仓库的整体性能。\\n\\n该任务已经完成100%的进度，因此可以跳过，不需要再进
行相关工作。根据进度排优先级，该任务的优先级为高，可以将其他任务优先安排在该任务之前进行处理。\\n\\n请注意，以
上是一个示例，实际任务的名称、进度和优先级可能会根据具体情况进行调整。"
```

```
  },
  {
    "role": "user",
    "content": "帮我生成一个周报"
  },
  {
```

```
    "role": "assistant",
    "content": "周报\\n\\n日期：2021年12月13日至2021年12月19日\\n\\n本周工作内容：\\n
1. 现行基于pg为贴源层的cdc采集切换至dbz采集：\\n    - 进行自测和修复工作，确保切换过程的顺利进行。
（进度：90%）\\n
2. 可视化CDC程序内嵌抽取全量mongo到pg的kettle流程：\\n    - 进行细节封装工作，确保内嵌流程的稳定运
行。（进度：70%）\\n
3. 支持xx项目中的cda映射问题：\\n    - 协助解决cda映射问题，确保数据准确性和一致性。（进度：100%）
\\n
4. 参与讨论IH0多schema版数据采集：\\n    - 提出建议和意见，为多schema版数据采集提供支持。（进度：
20%）\\n
5. xx项目doris-ods维护：\\n    - 进行doris-ods的维护工作，确保数据的及时更新和准确性。（进度：
50%）\\n\\n
```

```
    下周工作计划：\\n
1. 继续进行xx项目doris-ods的维护工作，确保数据的及时更新和准确性。\\n
2. 继续支持xx项目中的cda映射问题，协助解决问题，确保数据准确性和一致性。\\n
3. 参与讨论IH0多schema版数据采集，并提出建议和意见，为多schema版数据采集提供支持。\\n
4. 完成CDC可视化程序新增对应策略的工作，确保程序的功能完善和稳定性。\\n
5. 继续进行可视化CDC程序内嵌抽取全量mongo到pg的kettle流程的细节封装工作，确保内嵌流程的稳定运行。
\\n
6. 完成对现行基于pg为贴源层的cdc采集切换至dbz采集的自测和修复工作，确保切换过程的顺利进行。\\n
7. 继续进行xx项目ods的维护工作，确保数据的及时更新和准确性。\\n\\n
```

```
    备注：\\n本周工作进展顺利，已完成的任务进度符合预期。
    下周工作计划中的任务将根据实际情况进行调整和安排，以确保工作的顺利进行。\\n\\n
    以上为本周工作内容和下周工作计划，
    请查阅。如有任何问题或需要进一步讨论的事项，请及时与我联系。谢谢！"
  }
]
""
```

```
Launcher x index.ipynb x +
+ 🔍 📄 📄 📄 📄 📄 Code Python 3 (ipykernel)

def get_completion(prompt, model="gpt-3.5-turbo"):

    # 把用户输入加入消息历史
    messages.append({"role": "user", "content": prompt})

    response = client.chat.completions.create(
        model=model,
        messages=messages,
        temperature=0,
    )
    msg = response.choices[0].message.content

    # 把模型生成的回复加入消息历史。很重要，否则下次调用模型时，模型不知道上下文
    messages.append({"role": "assistant", "content": msg})
    return msg

get_completion("12-16的工作计划有没有写")
get_completion("帮我写一个，100%进度的任务跳过且按进度排一个优先级")
get_completion("帮我生成一个周报")
print_json(messages)

[
  {
    "role": "system",
    "content": "\n12-15完成工作:\n1、xx项目doris-ods维护: (50%)\n2、xx项目cda映射问题支持。(100%)\n3、IHO多schema版数据数据讨论:cdc可视化程序新增对应策略。(20%)\n\n12-14完成工作:\n1、可视化CDC程序内嵌抽取全量mongo到pg的kettle流程:确定联调请求参数;细节封装(100%)\n2、开启基于新架构doris cdc采集;doris建表类型修复(100%)\n3、xx项目ods维护。(100%)\n12-13计划:\n1、xx项目doris-ods维护.\n2、xx项目cda映射问题支持.\n\n12-13完成工作:\n1、现行基于pg为贴源层的cdc采集切换至dbz采集;衔接kettle导数。(90%)\n2、可视化CDC程序内嵌抽取全量mongo到pg的kettle流程:已跑通内嵌流程;细节封装中(70%)\n12-14计划:\n1、现行基于pg为贴源层的cdc采集切换至dbz采集;自测、修复。(90%)\n2、可视化CDC程序内嵌抽取全量mongo到pg的kettle流程:细节封装;确定联调请求参数(70%)\n"
  },
  {
    "role": "user",
    "content": "12-16的工作计划有没有写"
  },
  {
    "role": "assistant",
    "content": "抱歉，我之前没有写12-16的工作计划。以下是12-16的工作计划:\n\n1、继续进行xx项目doris-ods的维护工作;\n2、继续支持xx项目中的cda映射问题;\n3、参与讨论IHO多schema版数据数据，并提出建议;\n4、完成CDC可视化程序新增对应策略的工作;\n5、继续进行可视化CDC程序内嵌抽取全量mongo到pg的kettle流程的细节封装工作;\n6、完成对现行基于pg为贴源层的cdc采集切换至dbz采集的自测和修复工作;\n7、继续进行xx项目ods的维护工作.\n\n请注意，以上工作计划可能会根据实际情况进行调整。"
```

小结

个人原因，作业完成时间仓促，请老师批评和指正。

目前跑模型只是套用了一些模板来实现，包括openai api的调用都依赖于所学老师提供的OPENAI_BASE_URL和OPENAI_API_KEY，后续需要自己去注册一些license；

继续加强对prompt指令工程的理解和利用，更好的集成到业务系统中。

谢谢！