第八章 搭建一个带评估的端到端问答系统

在这一章节中，我们将会构建一个集成评估环节的完整问答系统。这个系统将会融合我们在前几节课中所学到的知识，并且加入了评估步骤。以下是该系统的核心操作流程：

1. 对用户的输入进行检验，验证其是否可以通过审核 API 的标准。
2. 若输入顺利通过审核，我们将进一步对产品目录进行搜索。
3. 若产品搜索成功，我们将继续寻找相关的产品信息。
4. 我们使用模型针对用户的问题进行回答。
5. 最后，我们会使用审核 API 对生成的回答进行再次的检验。

如果最终答案没有被标记为有害，那么我们将毫无保留地将其呈现给用户。

二、端到端实现问答系统

在我们的探索之旅中，我们将实现一个完整的问答系统，一种能理解并回应人类语言的人工智能。在这个过程中，我们将使用 OpenAI 的相关API并引用相关函数，来帮助我们快速搭建一个高效且精准的模型。然而，我们需要注意到，在中文的理解和处理方面，由于模型的特性，我们可能会偶尔遇到不理想的结果。在这种情况下，你可以多尝试几次，或者进行深入的研究，以找到更稳定的方法。

让我们先从一个函数开始，它的名称是 `process_user_message_ch`，该函数主要负责处理用户输入的信息。这个函数接收三个参数，用户的输入、所有的历史信息，以及一个表示是否需要调试的标志。在函数的内部，我们首先使用 OpenAI 的 Moderation API 来检查用户输入的合规性。如果输入被标记为不合规，我们将返回一个信息，告知用户请求不合规。在调试模式下，我们将打印出当前的进度。

接下来，我们利用 `utils_zh.find_category_and_product_only` 函数（详细请见附录代码）抽取出用户输入中的商品和对应的目录。然后，我们将抽取的信息转化为一个列表。
在获取到商品列表后，我们将查询这些商品的具体信息。之后，我们生成一个系统消息，设定一些约束，以确保我们的回应符合期望的标准。我们将生成的消息和历史信息一起送入 `get_completion_from_messages` 函数，得到模型的回应。之后，我们再次使用 Moderation API 检查模型的输出是否合规。如果输出不合规，我们将返回一个信息，告知无法提供该信息。

最后，我们让模型自我评估是否很好地回答了用户的问题。如果模型认为回答是满足要求的，我们则返回模型的回答；否则，我们会告知用户，他们将会被转接到人工客服进行进一步的帮助。

```
import openai
import utils_zh
from tool import get_completion_from_messages

'''
注意：限于模型对中文理解能力较弱，中文 Prompt 可能会随机出现不成功，可以多次运行；也非常欢迎同学探究更稳定的中文 Prompt
'''
def process_user_message_ch(user_input, all_messages, debug=True):
    """
    对用户信息进行预处理

    参数:
    user_input : 用户输入
    all_messages : 历史信息
    debug : 是否开启 DEBUG 模式,默认开启
    """
```

```
    # 分隔符
    delimiter = "```"

    # 第一步: 使用 OpenAI 的 Moderation API 检查用户输入是否合规或者是一个注入的 Prompt
    response = openai.Moderation.create(input=user_input)
    moderation_output = response["results"][0]

    # 经过 Moderation API 检查该输入不合规
    if moderation_output["flagged"]:
        print("第一步：输入被 Moderation 拒绝")
        return "抱歉，您的请求不合规"

    # 如果开启了 DEBUG 模式，打印实时进度
    if debug: print("第一步：输入通过 Moderation 检查")

    # 第二步：抽取出商品和对应的目录，类似于之前课程中的方法，做了一个封装
    category_and_product_response =
utils_zh.find_category_and_product_only(user_input,
utils_zh.get_products_and_category())
    #print(category_and_product_response)
    # 将抽取出来的字符串转化为列表
    category_and_product_list =
utils_zh.read_string_to_list(category_and_product_response)
    #print(category_and_product_list)

    if debug: print("第二步：抽取出商品列表")

    # 第三步：查找商品对应信息
    product_information =
utils_zh.generate_output_string(category_and_product_list)
    if debug: print("第三步：查找抽取出的商品信息")

    # 第四步：根据信息生成回答
    system_message = f"""
        您是一家大型电子商店的客户服务助理。\
        请以友好和乐于助人的语气回答问题，并提供简洁明了的答案。\
        请确保向用户提出相关的后续问题。
    """
    # 插入 message
    messages = [
        {'role': 'system', 'content': system_message},
        {'role': 'user', 'content': f"{delimiter}{user_input}{delimiter}"},
        {'role': 'assistant', 'content': f"相关商品信息:\n{product_information}"}
    ]
    # 获取 GPT3.5 的回答
    # 通过附加 all_messages 实现多轮对话
    final_response = get_completion_from_messages(all_messages + messages)
    if debug:print("第四步：生成用户回答")
    # 将该轮信息加入到历史信息中
    all_messages = all_messages + messages[1:]

    # 第五步：基于 Moderation API 检查输出是否合规
    response = openai.Moderation.create(input=final_response)
    moderation_output = response["results"][0]

    # 输出不合规
```

```
    if moderation_output["flagged"]:
        if debug: print("第五步：输出被 Moderation 拒绝")
        return "抱歉，我们不能提供该信息"

    if debug: print("第五步：输出经过 Moderation 检查")

    # 第六步：模型检查是否很好地回答了用户问题
    user_message = f"""
    用户信息: {delimiter}{user_input}{delimiter}
    代理回复: {delimiter}{final_response}{delimiter}

    回复是否足够回答问题
    如果足够，回答 Y
    如果不足够，回答 N
    仅回答上述字母即可
    """
    # print(final_response)
    messages = [
        {'role': 'system', 'content': system_message},
        {'role': 'user', 'content': user_message}
    ]
    # 要求模型评估回答
    evaluation_response = get_completion_from_messages(messages)
    # print(evaluation_response)
    if debug: print("第六步：模型评估该回答")

    # 第七步：如果评估为 Y，输出回答；如果评估为 N，反馈将由人工修正答案
    if "Y" in evaluation_response:  # 使用 in 来避免模型可能生成 Yes
        if debug: print("第七步：模型赞同了该回答.")
        return final_response, all_messages
    else:
        if debug: print("第七步：模型不赞成该回答.")
        neg_str = "很抱歉，我无法提供您所需的信息。我将为您转接到一位人工客服代表以获取进一步帮助。"
        return neg_str, all_messages

user_input = "请告诉我关于 smartx pro phone 和 the fotosnap camera 的信息。另外，请告诉我关于你们的tvs的情况。"
response,_ = process_user_message_ch(user_input,[])
print(response)
```

```
第一步：输入通过 Moderation 检查
第二步：抽取出商品列表
第三步：查找抽取出的商品信息
第四步：生成用户回答
第五步：输出经过 Moderation 检查
第六步：模型评估该回答
第七步：模型赞同了该回答.
关于SmartX ProPhone和FotoSnap相机的信息如下：

SmartX ProPhone：
- 品牌：SmartX
- 型号：SX-PP10
- 屏幕尺寸：6.1英寸
- 存储容量：128GB
```

- 相机：12MP双摄像头
- 网络：支持5G
- 保修：1年
- 价格：899.99美元

FotoSnap相机系列：
1. FotoSnap DSLR相机：
- 品牌：FotoSnap
- 型号：FS-DSLR200
- 传感器：24.2MP
- 视频：1080p
- 屏幕：3英寸LCD
- 可更换镜头
- 保修：1年
- 价格：599.99美元

2. FotoSnap无反相机：
- 品牌：FotoSnap
- 型号：FS-ML100
- 传感器：20.1MP
- 视频：4K
- 屏幕：3英寸触摸屏
- 可更换镜头
- 保修：1年
- 价格：799.99美元

3. FotoSnap即时相机：
- 品牌：FotoSnap
- 型号：FS-IC10
- 即时打印
- 内置闪光灯
- 自拍镜
- 电池供电
- 保修：1年
- 价格：69.99美元

关于我们的电视情况如下：

1. CineView 4K电视：
- 品牌：CineView
- 型号：CV-4K55
- 屏幕尺寸：55英寸
- 分辨率：4K
- HDR支持
- 智能电视功能
- 保修：2年
- 价格：599.99美元

2. CineView 8K电视：
- 品牌：

二、持续收集用户和助手消息

为了持续优化用户和助手的问答体验，我们打造了一个友好的可视化界面，以促进用户与助手之间的便捷互动。

```
# 调用中文 Prompt 版本
def collect_messages_ch(debug=True):
    """
    用于收集用户的输入并生成助手的回答

    参数:
    debug: 用于觉得是否开启调试模式
    """
    user_input = inp.value_input
    if debug: print(f"User Input = {user_input}")
    if user_input == "":
        return
    inp.value = ''
    global context
    # 调用 process_user_message 函数
    #response, context = process_user_message(user_input, context,
utils.get_products_and_category(),debug=True)
    response, context = process_user_message_ch(user_input, context, debug=False)
    # print(response)
    context.append({'role':'assistant', 'content':f"{response}"})
    panels.append(
        pn.Row('User:', pn.pane.Markdown(user_input, width=600)))
    panels.append(
        pn.Row('Assistant:', pn.pane.Markdown(response, width=600, style=
{'background-color': '#F6F6F6'})))

    return pn.Column(*panels) # 包含了所有的对话信息
```

```
import panel as pn  # 用于图形化界面
pn.extension()

panels = [] # collect display

# 系统信息
context = [ {'role':'system', 'content':"You are Service Assistant"} ]

inp = pn.widgets.TextInput( placeholder='Enter text here…')
button_conversation = pn.widgets.Button(name="Service Assistant")

interactive_conversation = pn.bind(collect_messages_ch, button_conversation)

dashboard = pn.Column(
    inp,
    pn.Row(button_conversation),
    pn.panel(interactive_conversation, loading_indicator=True, height=300),
)

dashboard
```

下图展示了该问答系统的运行实况：

通过监控该问答系统在更多输入上的回答效果，您可以修改步骤，提高系统的整体性能。

我们可能会察觉，在某些环节，我们的 Prompt 可能更好，有些环节可能完全可以省略，甚至，我们可能会找到更好的检索方法等等。

对于这个问题，我们将在接下来的章节中进行更深入的探讨。

三、英文版

1.1 端到端问答系统

```
import utils_en
import openai

def process_user_message(user_input, all_messages, debug=True):
    """
    对用户信息进行预处理

    参数:
    user_input : 用户输入
    all_messages : 历史信息
    debug : 是否开启 DEBUG 模式,默认开启
    """
    # 分隔符
    delimiter = "```"

    # 第一步: 使用 OpenAI 的 Moderation API 检查用户输入是否合规或者是一个注入的 Prompt
    response = openai.Moderation.create(input=user_input)
    moderation_output = response["results"][0]

    # 经过 Moderation API 检查该输入不合规
    if moderation_output["flagged"]:
        print("第一步：输入被 Moderation 拒绝")
        return "抱歉，您的请求不合规"

    # 如果开启了 DEBUG 模式，打印实时进度
    if debug: print("第一步：输入通过 Moderation 检查")

    # 第二步：抽取出商品和对应的目录，类似于之前课程中的方法，做了一个封装
```

```
    category_and_product_response =
utils_en.find_category_and_product_only(user_input,
utils_en.get_products_and_category())
    #print(category_and_product_response)
    # 将抽取出来的字符串转化为列表
    category_and_product_list =
utils_en.read_string_to_list(category_and_product_response)
    #print(category_and_product_list)

    if debug: print("第二步：抽取出商品列表")

    # 第三步：查找商品对应信息
    product_information =
utils_en.generate_output_string(category_and_product_list)
    if debug: print("第三步：查找抽取出的商品信息")

    # 第四步：根据信息生成回答
    system_message = f"""
    You are a customer service assistant for a large electronic store. \
    Respond in a friendly and helpful tone, with concise answers. \
    Make sure to ask the user relevant follow-up questions.
    """
    # 插入 message
    messages = [
        {'role': 'system', 'content': system_message},
        {'role': 'user', 'content': f"{delimiter}{user_input}{delimiter}"},
        {'role': 'assistant', 'content': f"Relevant product
information:\n{product_information}"}
    ]
    # 获取 GPT3.5 的回答
    # 通过附加 all_messages 实现多轮对话
    final_response = get_completion_from_messages(all_messages + messages)
    if debug:print("第四步：生成用户回答")
    # 将该轮信息加入到历史信息中
    all_messages = all_messages + messages[1:]

    # 第五步：基于 Moderation API 检查输出是否合规
    response = openai.Moderation.create(input=final_response)
    moderation_output = response["results"][0]

    # 输出不合规
    if moderation_output["flagged"]:
        if debug: print("第五步：输出被 Moderation 拒绝")
        return "抱歉，我们不能提供该信息"

    if debug: print("第五步：输出经过 Moderation 检查")

    # 第六步：模型检查是否很好地回答了用户问题
    user_message = f"""
    Customer message: {delimiter}{user_input}{delimiter}
    Agent response: {delimiter}{final_response}{delimiter}

    Does the response sufficiently answer the question?
    """
    messages = [
        {'role': 'system', 'content': system_message},
```

```
        {'role': 'user', 'content': user_message}
    ]
    # 要求模型评估回答
    evaluation_response = get_completion_from_messages(messages)
    if debug: print("第六步：模型评估该回答")

    # 第七步：如果评估为 Y，输出回答；如果评估为 N，反馈将由人工修正答案
    if "Y" in evaluation_response:  # 使用 in 来避免模型可能生成 Yes
        if debug: print("第七步：模型赞同了该回答.")
        return final_response, all_messages
    else:
        if debug: print("第七步：模型不赞成该回答.")
        neg_str = "很抱歉，我无法提供您所需的信息。我将为您转接到一位人工客服代表以获取进一步帮助。"
        return neg_str, all_messages

user_input = "tell me about the smartx pro phone and the fotosnap camera, the dslr one. Also what tell me about your tvs"
response,_ = process_user_message(user_input,[])
print(response)
```

```
第一步：输入通过 Moderation 检查
第二步：抽取出商品列表
第三步：查找抽取出的商品信息
第四步：生成用户回答
第五步：输出经过 Moderation 检查
第六步：模型评估该回答
第七步：模型赞同了该回答.
Sure! Here's some information about the SmartX ProPhone and the FotoSnap DSLR Camera:

1. SmartX ProPhone:
   - Brand: SmartX
   - Model Number: SX-PP10
   - Features: 6.1-inch display, 128GB storage, 12MP dual camera, 5G connectivity
   - Description: A powerful smartphone with advanced camera features.
   - Price: $899.99
   - Warranty: 1 year

2. FotoSnap DSLR Camera:
   - Brand: FotoSnap
   - Model Number: FS-DSLR200
   - Features: 24.2MP sensor, 1080p video, 3-inch LCD, interchangeable lenses
   - Description: Capture stunning photos and videos with this versatile DSLR camera.
   - Price: $599.99
   - Warranty: 1 year

Now, could you please let me know which specific TV models you are interested in?
```

2.1 持续收集用户和助手信息

```
def collect_messages_en(debug=False):
    """
```

```
    用于收集用户的输入并生成助手的回答

    参数:
    debug: 用于觉得是否开启调试模式
    """
    user_input = inp.value_input
    if debug: print(f"User Input = {user_input}")
    if user_input == "":
        return
    inp.value = ''
    global context
    # 调用 process_user_message 函数
    #response, context = process_user_message(user_input, context,
utils.get_products_and_category(),debug=True)
    response, context = process_user_message(user_input, context, debug=False)
    context.append({'role':'assistant', 'content':f"{response}"})
    panels.append(
        pn.Row('User:', pn.pane.Markdown(user_input, width=600)))
    panels.append(
        pn.Row('Assistant:', pn.pane.Markdown(response, width=600, style=
{'background-color': '#F6F6F6'})))

    return pn.Column(*panels) # 包含了所有的对话信息
```