第九章 评估（上）——存在一个简单的正确答案

在过去的章节里，我们向你展示了如何借助 LLM 构建应用程序，包括评估输入，处理输入，以及在呈现结果给用户之前进行最后的结果检查。然而，在构建出这样的系统后，我们应如何确知其运行状况呢？更甚者，当我们将其部署并让用户开始使用之后，我们又该如何追踪其表现，发现可能存在的问题，并持续优化它的回答质量呢？在本章里，我们将向你分享一些评估LLM输出的最佳实践。

构建基于LLM的应用程序与构建传统的监督学习应用程序有所不同。因为你可以快速地构建出基于LLM的应用程序，所以评估通常不从测试集开始。相反，你会逐渐地建立起一个测试样例的集合。

在传统的监督学习环境下，你需要收集训练集、开发集，或者留出交叉验证集，在整个开发过程中都会用到它们。然而，如果你能够在几分钟内定义一个 Prompt ，并在几小时内得到反馈结果，那么停下来收集一千个测试样本就会显得极为繁琐。因为现在，你可以在没有任何训练样本的情况下得到结果。

因此，在使用LLM构建应用程序时，你可能会经历以下流程：首先，你会在一到三个样本的小样本中调整 Prompt ，尝试使其在这些样本上起效。随后，当你对系统进行进一步测试时，可能会遇到一些棘手的例子，这些例子无法通过 Prompt 或者算法解决。这就是使用 ChatGPT API 构建应用程序的开发者所面临的挑战。在这种情况下，你可以将这些额外的几个例子添加到你正在测试的集合中，有机地添加其他难以处理的例子。最终，你会将足够多的这些例子添加到你逐步扩大的开发集中，以至于手动运行每一个例子以测试 Prompt 变得有些不便。然后，你开始开发一些用于衡量这些小样本集性能的指标，例如平均准确度。这个过程的有趣之处在于，如果你觉得你的系统已经足够好了，你可以随时停止，不再进行改进。实际上，很多已经部署的应用程序就在第一步或第二步就停下来了，而且它们运行得非常好。

值得注意的是，很多大型模型的应用程序没有实质性的风险，即使它没有给出完全正确的答案。但是，对于一些高风险的应用，如若存在偏见或不适当的输出可能对某人造成伤害，那么收集测试集、严格评估系统的性能，以及确保它在使用前能做对事情，就显得尤为重要。然而，如果你仅仅是用它来总结文章供自己阅读，而不是给其他人看，那么可能带来的风险就会较小，你可以在这个过程中早早地停止，而不必付出收集大规模数据集的巨大代价。

现在让我们进入更实际的应用阶段，将刚才所学的理论知识转化为实践。让我们一起研究一些真实的数据，理解其结构并使用我们的工具来分析它们。在我们的案例中，我们获取一组分类信息及其产品名称。让我们执行以下代码，以查看这些分类信息及其产品名称

```
import utils_zh

products_and_category = utils_zh.get_products_and_category()
products_and_category
```

```
{'电脑和笔记本': ['TechPro 超极本',
  'BlueWave 游戏本',
  'PowerLite Convertible',
  'TechPro Desktop',
  'BlueWave Chromebook'],
 '智能手机和配件': ['SmartX ProPhone'],
 '专业手机': ['MobiTech PowerCase',
  'SmartX MiniPhone',
  'MobiTech Wireless Charger',
  'SmartX EarBuds'],
 '电视和家庭影院系统': ['CineView 4K TV',
  'SoundMax Home Theater',
```

```
  'CineView 8K TV',
  'SoundMax Soundbar',
  'CineView OLED TV'],
 '游戏机和配件': ['GameSphere X',
  'ProGamer Controller',
  'GameSphere Y',
  'ProGamer Racing Wheel',
  'GameSphere VR Headset'],
 '音频设备': ['AudioPhonic Noise-Canceling Headphones',
  'WaveSound Bluetooth Speaker',
  'AudioPhonic True Wireless Earbuds',
  'WaveSound Soundbar',
  'AudioPhonic Turntable'],
 '相机和摄像机': ['FotoSnap DSLR Camera',
  'ActionCam 4K',
  'FotoSnap Mirrorless Camera',
  'ZoomMaster Camcorder',
  'FotoSnap Instant Camera']}
```

一、找出相关产品和类别名称

在我们进行开发时，通常需要处理和解析用户的输入。特别是在电商领域，可能会有各种各样的用户查询，例如："我想要最贵的电脑"。我们需要一个能理解这种语境，并能给出相关产品和类别的工具。下面这段代码实现的功能就是这样。

首先我们定义了一个函数`find_category_and_product_v1`，这个函数的主要目的是从用户的输入中解析出产品和类别。这个函数需要两个参数：`user_input`代表用户的查询，`products_and_category`是一个字典，其中包含了产品类型和对应产品的信息。

在函数的开始，我们定义了一个分隔符`delimiter`，用来在客户服务查询中分隔内容。随后，我们创建了一条系统消息。这条消息主要解释了系统的运作方式：用户会提供客户服务查询，查询会被分隔符`delimiter`分隔。系统会输出一个Python列表，列表中的每个对象都是Json对象。每个对象会包含'类别'和'名称'两个字段，分别对应产品的类别和名称。

我们创建了一个名为`messages`的列表，用来存储这些示例对话以及用户的查询。最后，我们使用`get_completion_from_messages`函数处理这些消息，返回处理结果。

通过这段代码，我们可以看到如何通过对话的方式理解和处理用户的查询，以提供更好的用户体验。

```
from tool import get_completion_from_messages

def find_category_and_product_v1(user_input,products_and_category):
    """
    从用户输入中获取到产品和类别

    参数:
    user_input：用户的查询
    products_and_category：产品类型和对应产品的字典
    """

    delimiter = "####"
    system_message = f"""
    您将提供客户服务查询。\
    客户服务查询将用{delimiter}字符分隔。
    输出一个 Python 列表，列表中的每个对象都是 Json 对象，每个对象的格式如下：
```

```
        '类别': <电脑和笔记本，智能手机和配件，电视和家庭影院系统，\
    游戏机和配件，音频设备，相机和摄像机中的一个>，
    以及
        '名称': <必须在下面允许的产品中找到的产品列表>

    其中类别和产品必须在客户服务查询中找到。
    如果提到了一个产品，它必须与下面允许的产品列表中的正确类别关联。
    如果没有找到产品或类别，输出一个空列表。

    根据产品名称和产品类别与客户服务查询的相关性，列出所有相关的产品。
    不要从产品的名称中假设任何特性或属性，如相对质量或价格。

    允许的产品以 JSON 格式提供。
    每个项目的键代表类别。
    每个项目的值是该类别中的产品列表。
    允许的产品：{products_and_category}

    """

    few_shot_user_1 = """我想要最贵的电脑。"""
    few_shot_assistant_1 = """
    [{'category': '电脑和笔记本', \
'products': ['TechPro 超极本', 'BlueWave 游戏本', 'PowerLite Convertible', 'TechPro
Desktop', 'BlueWave Chromebook']}]
    """

    messages =  [
    {'role':'system', 'content': system_message},
    {'role':'user', 'content': f"{delimiter}{few_shot_user_1}{delimiter}"},
    {'role':'assistant', 'content': few_shot_assistant_1 },
    {'role':'user', 'content': f"{delimiter}{user_input}{delimiter}"},
    ]
    return get_completion_from_messages(messages)
```

二、在一些查询上进行评估

对上述系统，我们可以首先在一些简单查询上进行评估：

```
# 第一个评估的查询
customer_msg_0 = f"""如果我预算有限，我可以买哪款电视？"""

products_by_category_0 = find_category_and_product_v1(customer_msg_0,
                                                      products_and_category)
print(products_by_category_0)
```

```
    [{'category': '电视和家庭影院系统', 'products': ['CineView 4K TV', 'SoundMax
Home Theater', 'CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV']}]
```

输出了正确回答。

```
customer_msg_1 = f"""我需要一个智能手机的充电器"""

products_by_category_1 = find_category_and_product_v1(customer_msg_1,
                                                      products_and_category)
print(products_by_category_1)
```

```
    [{'category': '智能手机和配件', 'products': ['MobiTech PowerCase', 'SmartX
MiniPhone', 'MobiTech Wireless Charger', 'SmartX EarBuds']}]
```

输出了正确回答。

```
customer_msg_2 = f"""
你们有哪些电脑？"""

products_by_category_2 = find_category_and_product_v1(customer_msg_2,
                                                      products_and_category)
products_by_category_2
```

```
" \n    [{'category': '电脑和笔记本', 'products': ['TechPro 超极本', 'BlueWave 游戏
本', 'PowerLite Convertible', 'TechPro Desktop', 'BlueWave Chromebook']}]"
```

输出回答正确，但格式有误。

```
customer_msg_3 = f"""
告诉我关于smartx pro手机和fotosnap相机的信息，那款DSLR的。
我预算有限，你们有哪些性价比高的电视推荐？"""

products_by_category_3 = find_category_and_product_v1(customer_msg_3,
                                                      products_and_category)
print(products_by_category_3)
```

```
    [{'category': '智能手机和配件', 'products': ['SmartX ProPhone']}, {'category':
'相机和摄像机', 'products': ['FotoSnap DSLR Camera']}]

    [{'category': '电视和家庭影院系统', 'products': ['CineView 4K TV', 'SoundMax
Home Theater', 'CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV']}]
```

它看起来像是输出了正确的数据，但没有按照要求的格式输出。这使得将其解析为 Python 字典列表更加困难。

三、更难的测试用例

接着，我们可以给出一些在实际使用中，模型表现不如预期的查询。

```
customer_msg_4 = f"""
告诉我关于CineView电视的信息，那款8K的，还有Gamesphere游戏机，X款的。
我预算有限，你们有哪些电脑？"""

products_by_category_4 =
find_category_and_product_v1(customer_msg_4,products_and_category)
print(products_by_category_4)
```

```
    [{'category': '电视和家庭影院系统', 'products': ['CineView 4K TV', 'SoundMax
Home Theater', 'CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV']}]
    [{'category': '游戏机和配件', 'products': ['GameSphere X', 'ProGamer
Controller', 'GameSphere Y', 'ProGamer Racing Wheel', 'GameSphere VR Headset']}]
    [{'category': '电脑和笔记本', 'products': ['TechPro 超极本', 'BlueWave 游戏本',
'PowerLite Convertible', 'TechPro Desktop', 'BlueWave Chromebook']}]
```

四、修改指令以处理难测试用例

综上，我们实现的最初版本在上述一些测试用例中表现不尽如人意。

为提升效果，我们在提示中添加了以下内容：不要输出任何不在 JSON 格式中的附加文本，并添加了第二个示例，使用用户和助手消息进行 few-shot 提示。

```
def find_category_and_product_v2(user_input,products_and_category):
    """
    从用户输入中获取到产品和类别

    添加：不要输出任何不符合 JSON 格式的额外文本。
    添加了第二个示例（用于 few-shot 提示），用户询问最便宜的计算机。
    在这两个 few-shot 示例中，显示的响应只是 JSON 格式的完整产品列表。

    参数：
    user_input：用户的查询
    products_and_category：产品类型和对应产品的字典
    """
    delimiter = "####"
    system_message = f"""
    您将提供客户服务查询。\
    客户服务查询将用{delimiter}字符分隔。
    输出一个 Python列表，列表中的每个对象都是 JSON 对象，每个对象的格式如下：
        '类别': <电脑和笔记本，智能手机和配件，电视和家庭影院系统，\
    游戏机和配件，音频设备，相机和摄像机中的一个>,
    以及
        '名称': <必须在下面允许的产品中找到的产品列表>
    不要输出任何不是 JSON 格式的额外文本。
    输出请求的 JSON 后，不要写任何解释性的文本。

    其中类别和产品必须在客户服务查询中找到。
    如果提到了一个产品，它必须与下面允许的产品列表中的正确类别关联。
    如果没有找到产品或类别，输出一个空列表。

    根据产品名称和产品类别与客户服务查询的相关性，列出所有相关的产品。
    不要从产品的名称中假设任何特性或属性，如相对质量或价格。

    允许的产品以 JSON 格式提供。
    每个项目的键代表类别。
    每个项目的值是该类别中的产品列表。
    允许的产品：{products_and_category}

    """

    few_shot_user_1 = """我想要最贵的电脑。你推荐哪款？"""
    few_shot_assistant_1 = """
```

```
    [{'category': '电脑和笔记本', \
'products': ['TechPro 超极本', 'BlueWave 游戏本', 'PowerLite Convertible', 'TechPro
Desktop', 'BlueWave Chromebook']}]
     """

    few_shot_user_2 = """我想要最便宜的电脑。你推荐哪款？ """
    few_shot_assistant_2 = """
    [{'category': '电脑和笔记本', \
'products': ['TechPro 超极本', 'BlueWave 游戏本', 'PowerLite Convertible', 'TechPro
Desktop', 'BlueWave Chromebook']}]
    """

    messages =  [
    {'role':'system', 'content': system_message},
    {'role':'user', 'content': f"{delimiter}{few_shot_user_1}{delimiter}"},
    {'role':'assistant', 'content': few_shot_assistant_1 },
    {'role':'user', 'content': f"{delimiter}{few_shot_user_2}{delimiter}"},
    {'role':'assistant', 'content': few_shot_assistant_2 },
    {'role':'user', 'content': f"{delimiter}{user_input}{delimiter}"},
    ]
    return get_completion_from_messages(messages)
```

五、在难测试用例上评估修改后的指令

我们可以在之前表现不如预期的较难测试用例上评估改进后系统的效果：

```
customer_msg_3 = f"""
告诉我关于smartx pro手机和fotosnap相机的信息，那款DSLR的。
另外，你们有哪些电视？"""

products_by_category_3 = find_category_and_product_v2(customer_msg_3,
                                                      products_and_category)
print(products_by_category_3)
```

```
    [{'category': '智能手机和配件', 'products': ['SmartX ProPhone']}, {'category':
'相机和摄像机', 'products': ['FotoSnap DSLR Camera', 'ActionCam 4K', 'FotoSnap
Mirrorless Camera', 'ZoomMaster Camcorder', 'FotoSnap Instant Camera']},
{'category': '电视和家庭影院系统', 'products': ['CineView 4K TV', 'SoundMax Home
Theater', 'CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV']}]
```

六、回归测试：验证模型在以前的测试用例上仍然有效

检查并修复模型以提高难以测试的用例效果，同时确保此修正不会对先前的测试用例性能造成负面影响。

```
customer_msg_0 = f"""如果我预算有限，我可以买哪款电视？"""

products_by_category_0 = find_category_and_product_v2(customer_msg_0,
                                                      products_and_category)
print(products_by_category_0)
```

```
    [{'category': '电视和家庭影院系统', 'products': ['CineView 4K TV', 'SoundMax
Home Theater', 'CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV']}]
```

七、收集开发集进行自动化测试

当我们的应用程序逐渐成熟，测试的重要性也随之增加。通常，当我们仅处理少量样本，手动运行测试并对结果进行评估是可行的。然而，随着开发集的增大，这种方法变得既繁琐又低效。此时，就需要引入自动化测试来提高我们的工作效率。下面将开始编写代码来自动化测试流程，可以帮助您提升效率并确保测试的准确率。

以下是一些用户问题的标准答案，用于评估 LLM 回答的准确度，与机器学习中的验证集的作用相当。

```
msg_ideal_pairs_set = [

    # eg 0
    {'customer_msg':"""如果我预算有限，我可以买哪种电视？""",
     'ideal_answer':{
        '电视和家庭影院系统':set(
            ['CineView 4K TV', 'SoundMax Home Theater', 'CineView 8K TV',
'SoundMax Soundbar', 'CineView OLED TV']
        )}
    },

    # eg 1
    {'customer_msg':"""我需要一个智能手机的充电器""",
     'ideal_answer':{
        '智能手机和配件':set(
            ['MobiTech PowerCase', 'MobiTech Wireless Charger', 'SmartX EarBuds']
        )}
    },
    # eg 2
    {'customer_msg':f"""你有什么样的电脑""",
     'ideal_answer':{
           '电脑和笔记本':set(
               ['TechPro 超极本', 'BlueWave 游戏本', 'PowerLite Convertible',
'TechPro Desktop', 'BlueWave Chromebook'
               ])
            }
    },

    # eg 3
    {'customer_msg':f"""告诉我关于smartx pro手机和fotosnap相机的信息，那款DSLR的。\
另外，你们有哪些电视？""",
     'ideal_answer':{
        '智能手机和配件':set(
            ['SmartX ProPhone']),
        '相机和摄像机':set(
            ['FotoSnap DSLR Camera']),
        '电视和家庭影院系统':set(
            ['CineView 4K TV', 'SoundMax Home Theater','CineView 8K TV',
'SoundMax Soundbar', 'CineView OLED TV'])
        }
    },
```

```
    # eg 4
    {'customer_msg':"""告诉我关于CineView电视，那款8K电视、\
     Gamesphere游戏机和X游戏机的信息。我的预算有限，你们有哪些电脑？""",
     'ideal_answer':{
        '电视和家庭影院系统':set(
            ['CineView 8K TV']),
        '游戏机和配件':set(
            ['GameSphere X']),
        '电脑和笔记本':set(
            ['TechPro Ultrabook', 'BlueWave Gaming Laptop', 'PowerLite
Convertible', 'TechPro Desktop', 'BlueWave Chromebook'])
        }
    },

    # eg 5
    {'customer_msg':f"""你们有哪些智能手机""",
     'ideal_answer':{
           '智能手机和配件':set(
               ['SmartX ProPhone', 'MobiTech PowerCase', 'SmartX MiniPhone',
'MobiTech Wireless Charger', 'SmartX EarBuds'
               ])
                    }
    },
    # eg 6
    {'customer_msg':f"""我预算有限。你能向我推荐一些智能手机吗？""",
     'ideal_answer':{
        '智能手机和配件':set(
            ['SmartX EarBuds', 'SmartX MiniPhone', 'MobiTech PowerCase', 'SmartX
ProPhone', 'MobiTech Wireless Charger']
        )}
    },

    # eg 7 # this will output a subset of the ideal answer
    {'customer_msg':f"""有哪些游戏机适合我喜欢赛车游戏的朋友？""",
     'ideal_answer':{
        '游戏机和配件':set([
            'GameSphere X',
            'ProGamer Controller',
            'GameSphere Y',
            'ProGamer Racing Wheel',
            'GameSphere VR Headset'
     ])}
    },
    # eg 8
    {'customer_msg':f"""送给我摄像师朋友什么礼物合适？""",
     'ideal_answer': {
        '相机和摄像机':set([
        'FotoSnap DSLR Camera', 'ActionCam 4K', 'FotoSnap Mirrorless Camera',
'ZoomMaster Camcorder', 'FotoSnap Instant Camera'
        ])}
    },

    # eg 9
    {'customer_msg':f"""我想要一台热水浴缸时光机""",
     'ideal_answer': []
    }
```

```
]
```

八、通过与理想答案比较来评估测试用例

我们通过以下函数`eval_response_with_ideal`来评估 LLM 回答的准确度，该函数通过将 LLM 回答与理想答案进行比较来评估系统在测试用例上的效果。

```
import json
def eval_response_with_ideal(response,
                             ideal,
                             debug=False):
    """
    评估回复是否与理想答案匹配

    参数:
    response: 回复的内容
    ideal: 理想的答案
    debug: 是否打印调试信息
    """
    if debug:
        print("回复: ")
        print(response)

    # json.loads() 只能解析双引号，因此此处将单引号替换为双引号
    json_like_str = response.replace("'",'"')

    # 解析为一系列的字典
    l_of_d = json.loads(json_like_str)

    # 当响应为空，即没有找到任何商品时
    if l_of_d == [] and ideal == []:
        return 1

    # 另外一种异常情况是，标准答案数量与回复答案数量不匹配
    elif l_of_d == [] or ideal == []:
        return 0

    # 统计正确答案数量
    correct = 0

    if debug:
        print("l_of_d is")
        print(l_of_d)

    # 对每一个问答对
    for d in l_of_d:

        # 获取产品和目录
        cat = d.get('category')
        prod_l = d.get('products')
        # 有获取到产品和目录
        if cat and prod_l:
            # convert list to set for comparison
```

```
            prod_set = set(prod_l)
            # get ideal set of products
            ideal_cat = ideal.get(cat)
            if ideal_cat:
                prod_set_ideal = set(ideal.get(cat))
            else:
                if debug:
                    print(f"没有在标准答案中找到目录 {cat}")
                    print(f"标准答案: {ideal}")
                continue

            if debug:
                print("产品集合：\n",prod_set)
                print()
                print("标准答案的产品集合：\n",prod_set_ideal)

            # 查找到的产品集合和标准的产品集合一致
            if prod_set == prod_set_ideal:
                if debug:
                    print("正确")
                correct +=1
            else:
                print("错误")
                print(f"产品集合: {prod_set}")
                print(f"标准的产品集合: {prod_set_ideal}")
                if prod_set <= prod_set_ideal:
                    print("回答是标准答案的一个子集")
                elif prod_set >= prod_set_ideal:
                    print("回答是标准答案的一个超集")

    # 计算正确答案数
    pc_correct = correct / len(l_of_d)

    return pc_correct
```

我们使用上述测试用例中的一个进行测试，首先看一下标准回答：

```
print(f'用户提问: {msg_ideal_pairs_set[7]["customer_msg"]}')
print(f'标准答案: {msg_ideal_pairs_set[7]["ideal_answer"]}')
```

```
用户提问: 有哪些游戏机适合我喜欢赛车游戏的朋友？
标准答案: {'游戏机和配件': {'ProGamer Racing Wheel', 'ProGamer Controller',
'GameSphere Y', 'GameSphere VR Headset', 'GameSphere X'}}
```

再对比 LLM 回答，并使用验证函数进行评分：

```
response = find_category_and_product_v2(msg_ideal_pairs_set[7]["customer_msg"],
                                        products_and_category)
print(f'回答: {response}')

eval_response_with_ideal(response,
                         msg_ideal_pairs_set[7]["ideal_answer"])
```

```
回答:
    [{'category': '游戏机和配件', 'products': ['GameSphere X', 'ProGamer
Controller', 'GameSphere Y', 'ProGamer Racing Wheel', 'GameSphere VR Headset']}]
```

```
1.0
```

可见该验证函数的打分是准确的。

九、在所有测试用例上运行评估，并计算正确的用例比例

下面我们来对测试用例中的全部问题进行验证，并计算 LLM 回答正确的准确率

> 注意：如果任何 API 调用超时，将无法运行

```python
import time

score_accum = 0
for i, pair in enumerate(msg_ideal_pairs_set):
    time.sleep(20)
    print(f"示例 {i}")

    customer_msg = pair['customer_msg']
    ideal = pair['ideal_answer']

    # print("Customer message",customer_msg)
    # print("ideal:",ideal)
    response = find_category_and_product_v2(customer_msg,
                                            products_and_category)


    # print("products_by_category",products_by_category)
    score = eval_response_with_ideal(response,ideal,debug=False)
    print(f"{i}: {score}")
    score_accum += score


n_examples = len(msg_ideal_pairs_set)
fraction_correct = score_accum / n_examples
print(f"正确比例为 {n_examples}: {fraction_correct}")
```

```
示例 0
0: 1.0
示例 1
错误
产品集合: {'SmartX ProPhone'}
标准的产品集合: {'MobiTech Wireless Charger', 'SmartX EarBuds', 'MobiTech
PowerCase'}
1: 0.0
示例 2
2: 1.0
示例 3
3: 1.0
示例 4
```

```
错误
产品集合: {'SoundMax Home Theater', 'CineView 8K TV', 'CineView 4K TV', 'CineView
OLED TV', 'SoundMax Soundbar'}
标准的产品集合: {'CineView 8K TV'}
回答是标准答案的一个超集
错误
产品集合: {'ProGamer Racing Wheel', 'ProGamer Controller', 'GameSphere Y',
'GameSphere VR Headset', 'GameSphere X'}
标准的产品集合: {'GameSphere X'}
回答是标准答案的一个超集
错误
产品集合: {'TechPro 超极本', 'TechPro Desktop', 'BlueWave Chromebook', 'PowerLite
Convertible', 'BlueWave 游戏本'}
标准的产品集合: {'TechPro Desktop', 'BlueWave Chromebook', 'TechPro Ultrabook',
'PowerLite Convertible', 'BlueWave Gaming Laptop'}
4: 0.0
示例 5
错误
产品集合: {'SmartX ProPhone'}
标准的产品集合: {'MobiTech Wireless Charger', 'SmartX EarBuds', 'SmartX MiniPhone',
'SmartX ProPhone', 'MobiTech PowerCase'}
回答是标准答案的一个子集
5: 0.0
示例 6
错误
产品集合: {'SmartX ProPhone'}
标准的产品集合: {'MobiTech Wireless Charger', 'SmartX EarBuds', 'SmartX MiniPhone',
'SmartX ProPhone', 'MobiTech PowerCase'}
回答是标准答案的一个子集
6: 0.0
示例 7
7: 1.0
示例 8
8: 1.0
示例 9
9: 1
正确比例为 10: 0.6
```

使用 Prompt 构建应用程序的工作流程与使用监督学习构建应用程序的工作流程非常不同。因此，我们认为这是需要记住的一件好事，当您正在构建监督学习模型时，会感觉到迭代速度快了很多。

如果您并未亲身体验，可能会惊叹于仅有手动构建的极少样本，就可以产生高效的评估方法。您可能会认为，仅有 10 个样本是不具备统计意义的。但当您真正运用这种方式时，您可能会对向开发集中添加一些复杂样本所带来的效果提升感到惊讶。这对于帮助您和您的团队找到有效的 Prompt 和有效的系统非常有帮助。

在本章中，输出可以被定量评估，就像有一个期望的输出一样，您可以判断它是否给出了这个期望的输出。在下一章中，我们将探讨如何在更加模糊的情况下评估我们的输出。即正确答案可能不那么明确的情况。

十、英文版

1. 找出产品和类别名称

```
import utils_en

products_and_category = utils_en.get_products_and_category()
products_and_category
```

```
{'Computers and Laptops': ['TechPro Ultrabook',
  'BlueWave Gaming Laptop',
  'PowerLite Convertible',
  'TechPro Desktop',
  'BlueWave Chromebook'],
 'Smartphones and Accessories': ['SmartX ProPhone',
  'MobiTech PowerCase',
  'SmartX MiniPhone',
  'MobiTech Wireless Charger',
  'SmartX EarBuds'],
 'Televisions and Home Theater Systems': ['CineView 4K TV',
  'SoundMax Home Theater',
  'CineView 8K TV',
  'SoundMax Soundbar',
  'CineView OLED TV'],
 'Gaming Consoles and Accessories': ['GameSphere X',
  'ProGamer Controller',
  'GameSphere Y',
  'ProGamer Racing Wheel',
  'GameSphere VR Headset'],
 'Audio Equipment': ['AudioPhonic Noise-Canceling Headphones',
  'WaveSound Bluetooth Speaker',
  'AudioPhonic True Wireless Earbuds',
  'WaveSound Soundbar',
  'AudioPhonic Turntable'],
 'Cameras and Camcorders': ['FotoSnap DSLR Camera',
  'ActionCam 4K',
  'FotoSnap Mirrorless Camera',
  'ZoomMaster Camcorder',
  'FotoSnap Instant Camera']}
```

```
def find_category_and_product_v1(user_input, products_and_category):
    """
    从用户输入中获取到产品和类别

    参数:
    user_input: 用户的查询
    products_and_category: 产品类型和对应产品的字典
    """

    # 分隔符
    delimiter = "####"
    # 定义的系统信息，陈述了需要 GPT 完成的工作
    system_message = f"""
    You will be provided with customer service queries. \
    The customer service query will be delimited with {delimiter} characters.
```

```
    Output a Python list of json objects, where each object has the following
format:
        'category': <one of Computers and Laptops, Smartphones and Accessories,
Televisions and Home Theater Systems, \
    Gaming Consoles and Accessories, Audio Equipment, Cameras and Camcorders>,
    AND
        'products': <a list of products that must be found in the allowed
products below>


    Where the categories and products must be found in the customer service
query.
    If a product is mentioned, it must be associated with the correct category in
the allowed products list below.
    If no products or categories are found, output an empty list.


    List out all products that are relevant to the customer service query based
on how closely it relates
    to the product name and product category.
    Do not assume, from the name of the product, any features or attributes such
as relative quality or price.

    The allowed products are provided in JSON format.
    The keys of each item represent the category.
    The values of each item is a list of products that are within that category.
    Allowed products: {products_and_category}


    """
    # 给出几个示例
    few_shot_user_1 = """I want the most expensive computer."""
    few_shot_assistant_1 = """
    [{'category': 'Computers and Laptops', \
'products': ['TechPro Ultrabook', 'BlueWave Gaming Laptop', 'PowerLite
Convertible', 'TechPro Desktop', 'BlueWave Chromebook']}]
    """

    messages =  [
    {'role':'system', 'content': system_message},
    {'role':'user', 'content': f"{delimiter}{few_shot_user_1}{delimiter}"},
    {'role':'assistant', 'content': few_shot_assistant_1 },
    {'role':'user', 'content': f"{delimiter}{user_input}{delimiter}"},
    ]
    return get_completion_from_messages(messages)
```

2. 在一些查询上进行评估

```
# 第一个评估的查询
customer_msg_0 = f"""Which TV can I buy if I'm on a budget?"""

products_by_category_0 = find_category_and_product_v1(customer_msg_0,
                                                      products_and_category)
print(products_by_category_0)
```

```
    [{'category': 'Televisions and Home Theater Systems', 'products': ['CineView 4K TV', 'SoundMax Home Theater', 'CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV']}]
```

```
# 第二个评估的查询
customer_msg_1 = f"""I need a charger for my smartphone"""

products_by_category_1 = find_category_and_product_v1(customer_msg_1,
                                                      products_and_category)
print(products_by_category_1)
```

```
    [{'category': 'Smartphones and Accessories', 'products': ['MobiTech PowerCase', 'MobiTech Wireless Charger', 'SmartX EarBuds']}]
```

```
# 第三个评估查询
customer_msg_2 = f"""
What computers do you have?"""

products_by_category_2 = find_category_and_product_v1(customer_msg_2,
                                                      products_and_category)
products_by_category_2
```

```
" \n    [{'category': 'Computers and Laptops', 'products': ['TechPro Ultrabook', 'BlueWave Gaming Laptop', 'PowerLite Convertible', 'TechPro Desktop', 'BlueWave Chromebook']}]"
```

```
# 第四个查询，更复杂
customer_msg_3 = f"""
tell me about the smartx pro phone and the fotosnap camera, the dslr one.
Also, what TVs do you have?"""

products_by_category_3 = find_category_and_product_v1(customer_msg_3,
                                                      products_and_category)
print(products_by_category_3)
```

```
    [{'category': 'Smartphones and Accessories', 'products': ['SmartX ProPhone']}, {'category': 'Cameras and Camcorders', 'products': ['FotoSnap DSLR Camera']}, {'category': 'Televisions and Home Theater Systems', 'products': ['CineView 4K TV', 'SoundMax Home Theater', 'CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV']}]
```

3. 更难的测试用例

```
customer_msg_4 = f"""
tell me about the CineView TV, the 8K one, Gamesphere console, the X one.
I'm on a budget, what computers do you have?"""

products_by_category_4 = find_category_and_product_v1(customer_msg_4,
                                                      products_and_category)
print(products_by_category_4)
```

```
    [{'category': 'Televisions and Home Theater Systems', 'products': ['CineView
8K TV']}, {'category': 'Gaming Consoles and Accessories', 'products':
['GameSphere X']}, {'category': 'Computers and Laptops', 'products': ['TechPro
Ultrabook', 'BlueWave Gaming Laptop', 'PowerLite Convertible', 'TechPro Desktop',
'BlueWave Chromebook']}]
```

4. 修改指令

```
def find_category_and_product_v2(user_input, products_and_category):
    """
    从用户输入中获取到产品和类别
    添加：不要输出任何不符合 JSON 格式的额外文本。
    添加了第二个示例（用于 few-shot 提示），用户询问最便宜的计算机。
    在这两个 few-shot 示例中，显示的响应只是 JSON 格式的完整产品列表。

    参数：
    user_input：用户的查询
    products_and_category：产品类型和对应产品的字典
    """
    delimiter = "####"
    system_message = f"""
    You will be provided with customer service queries. \
    The customer service query will be delimited with {delimiter} characters.
    Output a Python list of JSON objects, where each object has the following
format:
        'category': <one of Computers and Laptops, Smartphones and Accessories,
Televisions and Home Theater Systems, \
    Gaming Consoles and Accessories, Audio Equipment, Cameras and Camcorders>,
    AND
        'products': <a list of products that must be found in the allowed
products below>
    Do not output any additional text that is not in JSON format.
    Do not write any explanatory text after outputting the requested JSON.


    Where the categories and products must be found in the customer service
query.
    If a product is mentioned, it must be associated with the correct category in
the allowed products list below.
    If no products or categories are found, output an empty list.


    List out all products that are relevant to the customer service query based
on how closely it relates
    to the product name and product category.
```

```
    Do not assume, from the name of the product, any features or attributes such
as relative quality or price.

    The allowed products are provided in JSON format.
    The keys of each item represent the category.
    The values of each item is a list of products that are within that category.
    Allowed products: {products_and_category}


    """

    few_shot_user_1 = """I want the most expensive computer. What do you
recommend?"""
    few_shot_assistant_1 = """
    [{'category': 'Computers and Laptops', \
'products': ['TechPro Ultrabook', 'BlueWave Gaming Laptop', 'PowerLite
Convertible', 'TechPro Desktop', 'BlueWave Chromebook']}]
    """

    few_shot_user_2 = """I want the most cheapest computer. What do you
recommend?"""
    few_shot_assistant_2 = """
    [{'category': 'Computers and Laptops', \
'products': ['TechPro Ultrabook', 'BlueWave Gaming Laptop', 'PowerLite
Convertible', 'TechPro Desktop', 'BlueWave Chromebook']}]
    """

    messages =  [
    {'role':'system', 'content': system_message},
    {'role':'user', 'content': f"{delimiter}{few_shot_user_1}{delimiter}"},
    {'role':'assistant', 'content': few_shot_assistant_1 },
    {'role':'user', 'content': f"{delimiter}{few_shot_user_2}{delimiter}"},
    {'role':'assistant', 'content': few_shot_assistant_2 },
    {'role':'user', 'content': f"{delimiter}{user_input}{delimiter}"},
    ]
    return get_completion_from_messages(messages)
```

5. 进一步评估

```
customer_msg_3 = f"""
tell me about the smartx pro phone and the fotosnap camera, the dslr one.
Also, what TVs do you have?"""

products_by_category_3 = find_category_and_product_v2(customer_msg_3,
                                                      products_and_category)
print(products_by_category_3)
```

```
    [{'category': 'Smartphones and Accessories', 'products': ['SmartX
ProPhone']}, {'category': 'Cameras and Camcorders', 'products': ['FotoSnap DSLR
Camera']}, {'category': 'Televisions and Home Theater Systems', 'products':
['CineView 4K TV', 'SoundMax Home Theater', 'CineView 8K TV', 'SoundMax
Soundbar', 'CineView OLED TV']}]
```

6. 回归测试

```
customer_msg_0 = f"""Which TV can I buy if I'm on a budget?"""

products_by_category_0 = find_category_and_product_v2(customer_msg_0,
                                                      products_and_category)
print(products_by_category_0)
```

```
    [{'category': 'Televisions and Home Theater Systems', 'products': ['CineView 4K TV', 'SoundMax Home Theater', 'CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV']}]
```

7. 自动化测试

```
msg_ideal_pairs_set = [

    # eg 0
    {'customer_msg':"""Which TV can I buy if I'm on a budget?""",
     'ideal_answer':{
        'Televisions and Home Theater Systems':set(
            ['CineView 4K TV', 'SoundMax Home Theater', 'CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV']
        )}
    },

    # eg 1
    {'customer_msg':"""I need a charger for my smartphone""",
     'ideal_answer':{
        'Smartphones and Accessories':set(
            ['MobiTech PowerCase', 'MobiTech Wireless Charger', 'SmartX EarBuds']
        )}
    },
    # eg 2
    {'customer_msg':f"""What computers do you have?""",
     'ideal_answer':{
           'Computers and Laptops':set(
               ['TechPro Ultrabook', 'BlueWave Gaming Laptop', 'PowerLite Convertible', 'TechPro Desktop', 'BlueWave Chromebook'
               ])
                }
    },

    # eg 3
    {'customer_msg':f"""tell me about the smartx pro phone and \
    the fotosnap camera, the dslr one.\
    Also, what TVs do you have?""",
     'ideal_answer':{
        'Smartphones and Accessories':set(
            ['SmartX ProPhone']),
        'Cameras and Camcorders':set(
            ['FotoSnap DSLR Camera']),
        'Televisions and Home Theater Systems':set(
            ['CineView 4K TV', 'SoundMax Home Theater','CineView 8K TV', 'SoundMax Soundbar', 'CineView OLED TV'])
```

```
        }
    },

    # eg 4
    {'customer_msg':"""tell me about the CineView TV, the 8K one, Gamesphere
console, the X one.
I'm on a budget, what computers do you have?""",
     'ideal_answer':{
        'Televisions and Home Theater Systems':set(
            ['CineView 8K TV']),
        'Gaming Consoles and Accessories':set(
            ['GameSphere X']),
        'Computers and Laptops':set(
            ['TechPro Ultrabook', 'BlueWave Gaming Laptop', 'PowerLite
Convertible', 'TechPro Desktop', 'BlueWave Chromebook'])
        }
    },

    # eg 5
    {'customer_msg':f"""What smartphones do you have?""",
     'ideal_answer':{
           'Smartphones and Accessories':set(
               ['SmartX ProPhone', 'MobiTech PowerCase', 'SmartX MiniPhone',
'MobiTech Wireless Charger', 'SmartX EarBuds'
               ])
                    }
    },
    # eg 6
    {'customer_msg':f"""I'm on a budget.  Can you recommend some smartphones to
me?""",
     'ideal_answer':{
        'Smartphones and Accessories':set(
            ['SmartX EarBuds', 'SmartX MiniPhone', 'MobiTech PowerCase', 'SmartX
ProPhone', 'MobiTech Wireless Charger']
        )}
    },

    # eg 7 # this will output a subset of the ideal answer
    {'customer_msg':f"""What Gaming consoles would be good for my friend who is
into racing games?""",
     'ideal_answer':{
        'Gaming Consoles and Accessories':set([
            'GameSphere X',
            'ProGamer Controller',
            'GameSphere Y',
            'ProGamer Racing Wheel',
            'GameSphere VR Headset'
     ])}
    },
    # eg 8
    {'customer_msg':f"""What could be a good present for my videographer
friend?""",
     'ideal_answer': {
        'Cameras and Camcorders':set([
        'FotoSnap DSLR Camera', 'ActionCam 4K', 'FotoSnap Mirrorless Camera',
'ZoomMaster Camcorder', 'FotoSnap Instant Camera'
```

```
        ])}
    },

    # eg 9
    {'customer_msg':f"""I would like a hot tub time machine.""",
     'ideal_answer': []
    }

]
```

8. 与理想答案对比

```
import json
def eval_response_with_ideal(response,
                             ideal,
                             debug=False):
    """
    评估回复是否与理想答案匹配

    参数:
    response: 回复的内容
    ideal: 理想的答案
    debug: 是否打印调试信息
    """
    if debug:
        print("回复: ")
        print(response)

    # json.loads() 只能解析双引号，因此此处将单引号替换为双引号
    json_like_str = response.replace("'",'"')

    # 解析为一系列的字典
    l_of_d = json.loads(json_like_str)

    # 当响应为空，即没有找到任何商品时
    if l_of_d == [] and ideal == []:
        return 1

    # 另外一种异常情况是，标准答案数量与回复答案数量不匹配
    elif l_of_d == [] or ideal == []:
        return 0

    # 统计正确答案数量
    correct = 0

    if debug:
        print("l_of_d is")
        print(l_of_d)

    # 对每一个问答对
    for d in l_of_d:

        # 获取产品和目录
        cat = d.get('category')
```

```
        prod_l = d.get('products')
        # 有获取到产品和目录
        if cat and prod_l:
            # convert list to set for comparison
            prod_set = set(prod_l)
            # get ideal set of products
            ideal_cat = ideal.get(cat)
            if ideal_cat:
                prod_set_ideal = set(ideal.get(cat))
            else:
                if debug:
                    print(f"没有在标准答案中找到目录 {cat}")
                    print(f"标准答案: {ideal}")
                continue

            if debug:
                print("产品集合: \n",prod_set)
                print()
                print("标准答案的产品集合: \n",prod_set_ideal)

            # 查找到的产品集合和标准的产品集合一致
            if prod_set == prod_set_ideal:
                if debug:
                    print("正确")
                correct +=1
            else:
                print("错误")
                print(f"产品集合: {prod_set}")
                print(f"标准的产品集合: {prod_set_ideal}")
                if prod_set <= prod_set_ideal:
                    print("回答是标准答案的一个子集")
                elif prod_set >= prod_set_ideal:
                    print("回答是标准答案的一个超集")

    # 计算正确答案数
    pc_correct = correct / len(l_of_d)

    return pc_correct
```

```
print(f'用户提问: {msg_ideal_pairs_set[7]["customer_msg"]}')
print(f'标准答案: {msg_ideal_pairs_set[7]["ideal_answer"]}')
```

```
用户提问: What Gaming consoles would be good for my friend who is into racing
games?
标准答案: {'Gaming Consoles and Accessories': {'ProGamer Racing Wheel', 'ProGamer
Controller', 'GameSphere Y', 'GameSphere VR Headset', 'GameSphere X'}}
```

```
response = find_category_and_product_v2(msg_ideal_pairs_set[7]["customer_msg"],
                                        products_and_category)
print(f'回答: {response}')

eval_response_with_ideal(response,
                                msg_ideal_pairs_set[7]["ideal_answer"])
```

```
回答:
    [{'category': 'Gaming Consoles and Accessories', 'products': ['GameSphere X',
'ProGamer Controller', 'GameSphere Y', 'ProGamer Racing Wheel', 'GameSphere VR
Headset']}]
```

```
1.0
```

9. 计算正确比例

```
import time

score_accum = 0
for i, pair in enumerate(msg_ideal_pairs_set):
    time.sleep(20)
    print(f"示例 {i}")

    customer_msg = pair['customer_msg']
    ideal = pair['ideal_answer']

    # print("Customer message",customer_msg)
    # print("ideal:",ideal)
    response = find_category_and_product_v2(customer_msg,
                                                        products_and_category)


    # print("products_by_category",products_by_category)
    score = eval_response_with_ideal(response,ideal,debug=False)
    print(f"{i}: {score}")
    score_accum += score


n_examples = len(msg_ideal_pairs_set)
fraction_correct = score_accum / n_examples
print(f"正确比例为 {n_examples}: {fraction_correct}")
```

```
示例 0
0: 1.0
示例 1
错误
产品集合: {'MobiTech Wireless Charger', 'SmartX EarBuds', 'SmartX MiniPhone',
'SmartX ProPhone', 'MobiTech PowerCase'}
标准的产品集合: {'MobiTech Wireless Charger', 'SmartX EarBuds', 'MobiTech
PowerCase'}
回答是标准答案的一个超集
1: 0.0
示例 2
```

```
2: 1.0
示例 3
3: 1.0
示例 4
错误
产品集合: {'SoundMax Home Theater', 'CineView 8K TV', 'CineView 4K TV', 'CineView
OLED TV', 'SoundMax Soundbar'}
标准的产品集合: {'CineView 8K TV'}
回答是标准答案的一个超集
错误
产品集合: {'ProGamer Racing Wheel', 'ProGamer Controller', 'GameSphere Y',
'GameSphere VR Headset', 'GameSphere X'}
标准的产品集合: {'GameSphere X'}
回答是标准答案的一个超集
4: 0.3333333333333333
示例 5
5: 1.0
示例 6
6: 1.0
示例 7
7: 1.0
示例 8
8: 1.0
示例 9
9: 1
正确比例为 10: 0.8333333333333334
```