

## 17 | 让AI做决策，LangChain里的“中介”和“特工”

2023-04-14 徐文浩 来自北京

《AI大模型之美》



你好，我是徐文浩。

在🔗第 11 讲里，我为你讲解了如何把各种资料的内容向量化，然后通过 llama-index 建立对应的索引，实现对我们自己的文本资料的问答。而在过去的 3 讲里面，我们又深入了解了如何使用 Langchain。Langchain 能够便于我们把 AI 对语言的理解和组织能力、外部各种资料或者 SaaS 的 API，以及你自己撰写的代码整合到一起。通过对这些能力的整合，我们就可以通过自然语言完成更加复杂的任务了，而不仅仅是能闲聊。

不过，到目前为止。我们所有基于 ChatGPT 的应用，基本都是“单项技能”，比如前面关于“藤野先生”的问题，或者🔗上一讲里查询最新的天气或者通过 Python 算算术。本质上都是限制 AI 只针对我们预先索引的数据，或者实时搜索的数据进行回答。

**支持多种单项能力，让 AI 做个选择题**

但是，如果我们真的想要做一个能跑在生产环境上的 AI 聊天机器人，我们需要的不只一个单项技能。它应该针对你自己的数据有很多个不同的“单项技能”，就拿我比较熟悉的电商领域来说，我们至少需要这样三个技能。


1. 我们需要一个“导购咨询”的单项技能，能够查询自己商品库里的商品信息为用户做导购和推荐。
2. 然后需要一个“售中咨询”的单项技能，能够查询订单的物流轨迹，对买了东西还没有收到货的用户给出安抚和回复。
3. 最后还需要一个“FAQ”的单项技能，能够把整个电商网站的 FAQ 索引起来，当用户问到退货政策、运费、支付方式等问题的时候，我们可以从 FAQ 里面拿到对应的答案，回复给用户。

同时，对于这三个单项技能，AI 要能够自己判断什么时候该用什么样的技能。而不是需要人工介入，或者写一堆 if...else 的代码。

在学习了这么多讲的内容之后，你可以先想想，你有没有什么办法可以通过 ChatGPT 做到这些呢？直接一次性提供三个“单项技能”，还要在三个单项技能之间选择合适的技能，的确不是靠简单的几行代码或者 LLMChain 能够解决的。

但是我们可以采用一个在写大型系统的时候常用的思路，就是“分而治之”。对于每一个单项技能，我们都可以通过之前几讲学习的内容，把它们变成一个 LLMChain。然后，对于用户问的问题，我们不妨先问问 AI，让它告诉我们应该选用哪一个 LLMChain 来回答问题好了。

我们在下面就写了这样一段代码，通过提示语让 AI 做一个选择题。


 复制代码

```
1 import openai, os
2
3 openai.api_key = os.environ.get("OPENAI_API_KEY")
4
5 from langchain.prompts import PromptTemplate
6 from langchain.llms import OpenAIChat
7 from langchain.chains import LLMChain
8
9 llm = OpenAIChat(max_tokens=2048, temperature=0.5)
```

```
10 multiple_choice = """
11 请针对 >>> 和 <<< 中间的用户问题，选择一个合适的工具去回答她的问题。只要用A、B、C的选项字母告诉
12 如果你觉得都不合适，就选D。
13
14 >>>{question}<<<
15
16 我们有的工具包括：
17 A. 一个能够查询商品信息，为用户进行商品导购的工具
18 B. 一个能够查询订单信息，获得最新的订单情况的工具
19 C. 一个能够搜索商家的退换货政策、运费、物流时长、支付渠道、覆盖国家的工具
20 D. 都不合适
21 """
22 multiple_choice_prompt = PromptTemplate(template=multiple_choice, input_variables
23 choice_chain = LLMChain(llm=llm, prompt=multiple_choice_prompt, output_key="answer
```

对应的，我们可以试试问不同的问题，看看它能不能选择一个正确的工具。

问题 1:

 复制代码

```
1 question = "我想买一件衣服，但是不知道哪个款式好看，你能帮我推荐一下吗？"
2 print(choice_chain(question))
```

输出结果:

 复制代码


```
1 {'question': '我想买一件衣服，但是不知道哪个款式好看，你能帮我推荐一下吗？', 'answer': '\n\
```

问题 2:

 复制代码


```
1 question = "我有一张订单，订单号是 2022ABCDE，一直没有收到，能麻烦帮我查一下吗？"
2 print(choice_chain(question))
```

输出结果:

 复制代码


```
1 {'question': '我有一张订单，订单号是 2022ABCDE，一直没有收到，能麻烦帮我查一下吗？', 'answer': '好的，请稍等，我帮您查询一下。'}
```

问题 3:

 复制代码


```
1 question = "请问你们的货，能送到三亚吗？大概需要几天？"  
2 print(choice_chain(question))
```

输出结果:

 复制代码


```
1 {'question': '请问你们的货，能送到三亚吗？大概需要几天？', 'answer': '\n\nC. 一个能够搜索并返回结果的 API。'}
```

问题 4:

 复制代码

```
1 question = "今天天气怎么样？"  
2 print(choice_chain(question))
```

输出结果:

 复制代码

```
1 {'question': '今天天气怎么样？', 'answer': '\n\nD. 都不合适，因为这个问题需要的是天气预报信息。'}
```


可以看到，我们试了四个问题，ChatGPT 都给出了正确的答案。在拿到答案之后，你可以直接再通过一个 TransformChain，去匹配返回结果的前缀，看看是 A、B、C、D 中的哪一个，再来决定后面可以去调用哪个 LLMChain。

## Langchain 里面的中介与特工：Agent

这样一个“分治法”的思路，你在真实的业务场景中一定会遇到的。无论是哪行哪业的客服聊天机器人，其实都会有能够直接通过资料库就回答的用户问题，也会有和用户自己或者公司产品相关的信息，需要通过检索的方式来提供。所以，这样一个“先做一个选择题”的思路，Langchain 就把它发扬光大了，建立起了 Agent 这个抽象概念。

Agent 翻译成中文，有两个意思。一个叫做代理人，比如在美国你买房子、租房子，都要通过 Real Estate Agent，也就是“房产代理”，其实就是我们这里说的“中介”。另一个意思，叫做“特工”，这是指 Agent 是有自主行动能力的，它可以根据你提出的要求，直接使用提供的工具采取行动。它不只是做完选择题就完事儿了，而是直接拿起选中的工具进行下一步的行动。Langchain 的 Agent 其实这两个意思都包含，可以说名字取得是非常得当了。

下面我们来看看上面这个例子，我们怎么通过 Langchain 提供的 Agent 直接采取行动来解决问题。

 复制代码

```
1 from langchain.agents import initialize_agent, Tool
2 from langchain.llms import OpenAI
3
4 llm = OpenAI(temperature=0)
5
6 def search_order(input: str) -> str:
7     return "订单状态: 已发货; 发货日期: 2023-01-01; 预计送达时间: 2023-01-10"
8
9 def recommend_product(input: str) -> str:
10    return "红色连衣裙"
11
12 def faq(input: str) -> str:
13    return "7天无理由退货"
14
15 tools = [
16     Tool(
17         name = "Search Order", func=search_order,
18         description="useful for when you need to answer questions about customers
19     ),
20     Tool(name="Recommend Product", func=recommend_product,
21         description="useful for when you need to answer questions about product
22     ),
23     Tool(name="FAQ", func=faq,
24         description="useful for when you need to answer questions about shopping
25     )
26 ]
```

```
27 agent = initialize_agent(tools, llm, agent="zero-shot-react-description", verbose
28
```

这段代码由三个部分组成。

1. 首先，我们定义了三个函数，分别叫做 `search_order`、`recommend_product` 以及 `faq`。它们的输入都是一个字符串，输出是我们写好的对于问题的回答。
2. 然后，我们针对这三个函数，创建了一个 `Tool` 对象的数组，把这三个函数分别封装在了三个 `Tool` 对象里面。每一个 `Tool` 的对象，在函数之外，还定义了一个名字，并且定义了 `Tool` 的 `description`。这个 `description` 就是告诉 AI，这个 `Tool` 是干什么用的。就像这一讲一开始的那个例子一样，AI 会根据问题以及这些描述来做选择题。
3. 最后，我们创建了一个 `agent` 对象，指定它会用哪些 `Tools`、`LLM` 对象以及 `agent` 的类型。在 `agent` 的类型这里，我们选择了 `zero-shot-react-description`。这里的 `zero-shot` 就是指我们在课程一开始就讲过的“零样本分类”，也就是不给 AI 任何例子，直接让它根据自己的推理能力来做决策。而 `react description`，指的是根据你对于 `Tool` 的描述（`description`）进行推理（`Reasoning`）并采取行动（`Action`）。

这里的这个 `ReAct`，并不是来自 Facebook 的前端框架的名字，而是来自一篇 Google Brain 的论文 [ReAct: Synergizing Reasoning and Acting in Language Models](#)。有兴趣的话，你可以去阅读一下，了解具体的原理和思路。

在有了这个 `agent` 之后，我们不妨尝试一下，直接对着这个 `Agent`，来重新问一遍刚才的三个问题。


问题 1：

 复制代码

```
1 question = "我想买一件衣服，但是不知道哪个款式好看，你能帮我推荐一下吗？"
2 result = agent.run(question)
3 print(result)
```


输出结果：



 复制代码


```
1 > Entering new AgentExecutor chain...
2 I need to recommend a product.
3 Action: Recommend Product
4 Action Input: Clothes
5 Observation: 红色连衣裙
6 Thought: I now know the final answer.
7 Final Answer: 我推荐红色连衣裙。
```

## 问题 2:

 复制代码

```
1 question = "我有一张订单，订单号是 2022ABCDE，一直没有收到，能麻烦帮我查一下吗？"
2 result = agent.run(question)
3 print(result)
```

## 输出结果:

 复制代码


```
1 > Entering new AgentExecutor chain...
2 I need to find out the status of the order
3 Action: Search Order
4 Action Input: 2022ABCDE
5 Observation: 订单状态: 已发货; 发货日期: 2023-01-01; 预计送达时间: 2023-01-10
6 Thought: I now know the final answer
7 Final Answer: 您的订单 2022ABCDE 已发货, 预计将于2023-01-10送达。
8 > Finished chain.
9 您的订单 2022ABCDE 已发货, 预计将于2023-01-10送达。
```

## 问题 3:

 复制代码

```
1 question = "请问你们的货，能送到三亚吗？大概需要几天？"
2 result = agent.run(question)
3 print(result)
```

输出结果：

 复制代码

```
1
2 > Entering new AgentExecutor chain...
3 I need to find out the shipping policy and delivery time
4 Action: FAQ
5 Action Input: Shipping policy and delivery time
6 Observation: 7天无理由退货
7 Thought: I need to find out the delivery time
8 Action: FAQ
9 Action Input: Delivery time
10 Observation: 7天无理由退货
11 Thought: I need to find out if we can deliver to Sanya
12 Action: FAQ
13 Action Input: Delivery to Sanya
14 Observation: 7天无理由退货
15 Thought: I now know the final answer
16 Final Answer: 我们可以把货送到三亚，大概需要7天。
17 > Finished chain.
18 我们可以把货送到三亚，大概需要7天。
```

因为在代码里面，我们把 Agent 的 Verbose 模式打开了，所以在输出结果里面，你可以直接看到 Agent 整个思考的日志。从这里，你会发现几个有意思的现象。

第一个，是 Agent 每一步的操作，可以分成 5 个步骤，分别是 Action、Action Input、Observation、Thought，最后输出一个 Final Answer。

1. Action，就是根据用户的输入，选择应该选取哪一个 Tool，然后行动。
2. Action Input，就是根据需要使用到的 Tool，从用户的输入里提取出相关的内容，可以输入到 Tool 里面。
3. Observation，就是观察通过使用 Tool 得到的一个输出结果。
4. Thought，就是再看一眼用户的输入，判断一下该怎么做。
5. Final Answer，就是 Thought 在看到 Observation 之后，给出的最终输出。



第二个，就是我们最后那个“货需要几天送到三亚”的问题，没有遵循上面的 5 个步骤，而是在第 4 步 Thought 之后，重新回到了 Action。并且在这样反复三次之后，才不得已强行回答了问题。但是给出的答案，其实并不一定准确，因为我们的回答里面并没有说能不能送到三亚。

这一整个过程，其实也是通过一段 Prompt 来实现的，你可以去看一下 Langchain 源码里，[🔗 mrkl 对应的 Prompt 的源代码](#)。

 复制代码


```
1 # flake8: noqa
2 PREFIX = """Answer the following questions as best you can. You have access to th
3 FORMAT_INSTRUCTIONS = """Use the following format:
4
5 Question: the input question you must answer
6 Thought: you should always think about what to do
7 Action: the action to take, should be one of [{tool_names}]
8 Action Input: the input to the action
9 Observation: the result of the action
10 ... (this Thought/Action/Action Input/Observation can repeat N times)
11 Thought: I now know the final answer
12 Final Answer: the final answer to the original input question"""
13 SUFFIX = """Begin!
14
15 Question: {input}
16 Thought:{agent_scratchpad}"""
```

其实它就是把一系列的工具名称和对应的描述交给了 OpenAI，让它根据用户输入的需求，选取对应的工具，然后提取用户输入中和用户相关的信息。本质上，只是我们上面让 AI 做选择题的一种扩展而已。

## 通过 max\_iterations 限制重试次数


前面这个反复思考 3 次，其实是 Agent 本身的功能。因为实际很多逻辑处理，现在都是通过 AI 的大语言模型这个黑盒子自动进行的，有时候也不一定准。所以 AI 会在 Thought 的时候，看一下回答得是否靠谱，如果不靠谱的话，它会想一个办法重试。如果你希望 AI 不要不断重试，也不要强行回答，在觉得不靠谱的时候，试个一两次就停下来。那么，你在创建

Agent 的时候，设置 max\_iterations 这个参数就好了。下面我们就把参数设置成 2，看看效果会是怎么样的。

 复制代码

```
1 agent = initialize_agent(tools, llm, agent="zero-shot-react-description", max_ite
2 question = "请问你们的货，能送到三亚吗？大概需要几天？"
3 result = agent.run(question)
4 print("===")
5 print(result)
6 print("===")
```

输出结果：

 复制代码

```
1 > Entering new AgentExecutor chain...
2 I need to find out the shipping policy
3 Action: FAQ
4 Action Input: Shipping policy
5 Observation: 7天无理由退货
6 Thought: I need to find out the shipping time
7 Action: FAQ
8 Action Input: Shipping time
9 Observation: 7天无理由退货
10 Thought:
11 > Finished chain.
12 ===
13 Agent stopped due to max iterations.
14 ===
```

可以看到，这个时候，AI 重试了两次就不再重试。并且，也没有强行给出一个回答，而是告诉你，Agent 因为 max iterations 的设置而中止了。这样，你可以把 AI 回答不上来的问题，切换给人工客服回答。

## 通过 VectorDBQA 让 Tool 支持问答

当然，这么简单的问题我们完全可以让 AI 答上来。现在答不上来的原因是无论我们问什么问题，FQA 这个工具的回答都是 7 天无理由退货。而正确的方式其实也有，我们可以直接使用

🔗 第 15 讲介绍的 VectorDBQA 这个 LLMChain, 把它也封装成一个 Tool。

我们先把 🔗 第 15 讲对应的代码搬运过来。

📄 复制代码

```
1 from langchain.embeddings.openai import OpenAIEmbeddings
2 from langchain.vectorstores import FAISS
3 from langchain.text_splitter import SpacyTextSplitter
4 from langchain import OpenAI, VectorDBQA
5 from langchain.document_loaders import TextLoader
6
7 llm = OpenAI(temperature=0)
8 loader = TextLoader('./data/ecommerce_faq.txt')
9 documents = loader.load()
10 text_splitter = SpacyTextSplitter(chunk_size=256, pipeline="zh_core_web_sm")
11 texts = text_splitter.split_documents(documents)
12
13 embeddings = OpenAIEmbeddings()
14 docsearch = FAISS.from_documents(texts, embeddings)
15
16 faq_chain = VectorDBQA.from_chain_type(llm=llm, vectorstore=docsearch, verbose=Tr
```

然后, 把这 LLMChain 的 run 方法包装到一个 Tool 里面。

📄 复制代码

```
1 from langchain.agents import tool
2
3 @tool("FAQ")
4 def faq(input: str) -> str:
5     """useful for when you need to answer questions about shopping policies, lik
6     return faq_chain.run(input)
7
8 tools = [
9     Tool(
10         name = "Search Order", func=search_order,
11         description="useful for when you need to answer questions about customers
12     ),
13     Tool(name="Recommend Product", func=recommend_product,
14         description="useful for when you need to answer questions about product
15     ),
16     faq
17 ]
```

```
18 agent = initialize_agent(tools, llm, agent="zero-shot-react-description", verbose
19
```


这里，我们对 Tool 的写法做了一些小小的改变，使得代码更加容易维护了。我们通过 @tool 这个 Python 的 decorator 功能，将 FAQ 这个函数直接变成了 Tool 对象，这可以减少我们每次创建 Tools 的时候都要指定 name 和 description 的工作。

接着，我们再通过 Agent 来运行一下刚才的问题，一样能够得到正确的答案。

 复制代码

```
1 question = "请问你们的货，能送到三亚吗？大概需要几天？"
2 result = agent.run(question)
3 print(result)
```


输出结果：

 复制代码

```
1 > Entering new AgentExecutor chain...
2 I need to find out the shipping policy and delivery time.
3 Action: FAQ
4 Action Input: shipping policy and delivery time
5 > Entering new VectorDBQA chain...
6 > Finished chain.
7 Observation: 一般情况下，大部分城市的订单在2-3个工作日内送达，偏远地区可能需要5-7个工作日。
8 Thought: I now know the final answer.
9 Final Answer: 一般情况下，大部分城市的订单在2-3个工作日内送达，偏远地区可能需要5-7个工作日。
10 > Finished chain.
11 一般情况下，大部分城市的订单在2-3个工作日内送达，偏远地区可能需要5-7个工作日。具体送货时间可能因
```


对于商品的推荐，我们可以如法炮制，也把对应的商品信息，存到 VectorStore 里，然后通过 **先搜索后问答** 的方式来解决。对应的数据同样由 ChatGPT 出品，代码和上面的 FAQ 基本类似，我就不再一一重复了。

重新构建 Agent：

 复制代码


```
1 from langchain.text_splitter import CharacterTextSplitter
2 from langchain.document_loaders import CSVLoader
3
4 product_loader = CSVLoader('./data/ecommerce_products.csv')
5 product_documents = product_loader.load()
6 product_text_splitter = CharacterTextSplitter(chunk_size=1024, separator="\n")
7 product_texts = product_text_splitter.split_documents(product_documents)
8 product_search = FAISS.from_documents(product_texts, OpenAIEmbeddings())
9 product_chain = VectorDBQA.from_chain_type(llm=llm, vectorstore=product_search, v
10
11 @tool("FAQ")
12 def faq(input: str) -> str:
13     """useful for when you need to answer questions about shopping policies, lik
14     return faq_chain.run(input)
15
16 @tool("Recommend Product")
17 def recommend_product(input: str) -> str:
18     """useful for when you need to search and recommend products and recommend i
19     return product_chain.run(input)
20
21 tools = [
22     Tool(
23         name = "Search Order", func=search_order,
24         description="useful for when you need to answer questions about customers
25     ),
26     recommend_product, faq]
27
28 agent = initialize_agent(tools, llm, agent="zero-shot-react-description", verbose
29
```

询问 Agent 问题:

 复制代码

```
1 question = "我想买一件衣服，想要在春天去公园穿，但是不知道哪个款式好看，你能帮我推荐一下吗？"
2 answer = agent.run(question)
3 print(answer)
```


输出结果:

 复制代码

```
1 > Entering new AgentExecutor chain...
2 I need to recommend a product to the user.
3 Action: Recommend Product
4 Action Input: Clothing for park in spring
5 > Entering new VectorDBQA chain...
6 > Finished chain.
7 Observation: 休闲、简约风格的长款风衣, 休闲、运动风格的长款卫衣, 清新、甜美风格的长袖连衣裙都
8 Thought: I now know the final answer.
9 Final Answer: 休闲、简约风格的长款风衣, 休闲、运动风格的长款卫衣, 清新、甜美风格的长袖连衣裙都
10 > Finished chain.
11 休闲、简约风格的长款风衣, 休闲、运动风格的长款卫衣, 清新、甜美风格的长袖连衣裙都可以在春天去公园逛
```

## 优化 Prompt, 让 AI 不要胡思乱想

对于订单查询, 使用向量检索就不太合适了, 我们直接拿着订单号去数据库里查就好了。不过我们不是一个 Python 编程课, 我就不为你演示怎么用 Python 写 SQL 了。我们偷个懒, 就在对应的函数里造几条数据, 根据用户输入的订单号不同, 就返回不同的订单状态, 找不到的话, 就告诉用户找不到订单就好。

 复制代码


```
1 import json
2
3 ORDER_1 = "20230101ABC"
4 ORDER_2 = "20230101EFG"
5
6 ORDER_1_DETAIL = {
7     "order_number": ORDER_1,
8     "status": "已发货",
9     "shipping_date" : "2023-01-03",
10    "estimated_delivered_date": "2023-01-05",
11 }
12
13 ORDER_2_DETAIL = {
14     "order_number": ORDER_2,
15     "status": "未发货",
16     "shipping_date" : None,
17     "estimated_delivered_date": None,
18 }
19
20 import re
21
22 @tool("Search Order")
23 def search_order(input:str)->str:
24     """useful for when you need to answer questions about customers orders"""
```

```

25     if input.strip() == ORDER_1:
26         return json.dumps(ORDER_1_DETAIL)
27     elif input.strip() == ORDER_2:
28         return json.dumps(ORDER_2_DETAIL)
29     else:
30         return f"对不起, 根据{input}没有找到您的订单"
31
32 tools = [search_order, recommend_product, faq]
33 agent = initialize_agent(tools, llm=OpenAI(temperature=0), agent="zero-shot-react

```

然后，我们就可以试着来让 Agent 帮我们查一下订单号。

 复制代码

```

1 question = "我有一张订单，订单号是 2022ABCDE，一直没有收到，能麻烦帮我查一下吗？"
2 answer = agent.run(question)
3 print(answer)

```

输出结果：

 复制代码

```

1
2 > Entering new AgentExecutor chain...
3 I need to find out the status of the order.
4 Action: Search Order
5 Action Input: 2022ABCDE
6 Observation: 对不起, 根据2022ABCDE没有找到您的订单
7 Thought: I need to find out more information about the order.
8 Action: Search Order
9 Action Input: 2022ABCDE
10 Observation: 对不起, 根据2022ABCDE没有找到您的订单
11 Thought: I need to contact customer service for more information.
12 Action: FAQ
13 Action Input: 订单查询
14 > Entering new VectorDBQA chain...
15 > Finished chain.
16 Observation: 要查询订单，请登录您的帐户，然后点击“我的订单”页面。在此页面上，您可以查看所有订
17 Thought: I now know the final answer.
18 Final Answer: 要查询订单，请登录您的帐户，然后点击“我的订单”页面。在此页面上，您可以查看所有订
19 > Finished chain.
20 要查询订单，请登录您的帐户，然后点击“我的订单”页面。在此页面上，您可以查看所有订单及其当前状态。

```



结果稍稍让人有些意外，我们输入了一个不存在的订单号，我们原本期望，AI 能够告诉我们订单号找不到。但是，它却是在发现回复是找不到订单的时候，重复调用 OpenAI 的思考策略，并最终尝试从 FAQ 里拿一个查询订单的问题来敷衍用户。这并不是我们想要的，这也是以前很多“人工智障”类型的智能客服常常会遇到的问题，所以我们还是想个办法解决它。

解决的方法也不复杂，我们只需要调整一下 search\_order 这个 Tool 的提示语。通过这个提示语，Agent 会知道，这个工具就应该在找不到订单的时候，告诉用户找不到订单或者请它再次确认。这个时候，它就会根据这个答案去回复用户。下面是对应修改运行后的结果。

 复制代码

```
1 import re
2
3 @tool("Search Order")
4 def search_order(input:str)->str:
5     """一个帮助用户查询最新订单状态的工具，并且能处理以下情况：
6     1. 在用户没有输入订单号的时候，会询问用户订单号
7     2. 在用户输入的订单号查询不到的时候，会让用户二次确认订单号是否正确"""
8     pattern = r"\d+[A-Z]+"
9     match = re.search(pattern, input)
10
11     order_number = input
12     if match:
13         order_number = match.group(0)
14     else:
15         return "请问您的订单号是多少？"
16     if order_number == ORDER_1:
17         return json.dumps(ORDER_1_DETAIL)
18     elif order_number == ORDER_2:
19         return json.dumps(ORDER_2_DETAIL)
20     else:
21         return f"对不起，根据{input}没有找到您的订单"
22
23 tools = [search_order, recommend_product, faq]
24 agent = initialize_agent(tools, llm=OpenAI(temperature=0), agent="zero-shot-react"
25
26 question = "我有一张订单，订单号是 2022ABCDE，一直没有收到，能麻烦帮我查一下吗？"
27 answer = agent.run(question)
28 print(answer)
```

输出结果：

```

1 > Entering new AgentExecutor chain...
2 我需要查询订单状态
3 Action: Search Order
4 Action Input: 2022ABCDE
5 Observation: 对不起, 根据2022ABCDE没有找到您的订单
6 Thought: 我需要再次确认订单号是否正确
7 Action: Search Order
8 Action Input: 2022ABCDE
9 Observation: 对不起, 根据2022ABCDE没有找到您的订单
10 Thought: 我现在知道最终答案
11 Final Answer: 对不起, 根据您的输入的单号2022ABCDE没有找到您的订单, 请您再次确认订单号是否正确
12 > Finished chain.
13 对不起, 根据您的输入的单号2022ABCDE没有找到您的订单, 请您再次确认订单号是否正确。

```

## 通过多轮对话实现订单查询

看起来, 我们的客服聊天机器人已经搞定了。但是, 其实我们还有几个可以优化的空间。

1. 我们应该支持多轮聊天。因为用户不一定是在第一轮提问的时候, 就给出了自己的订单号。
2. 我们其实可以直接让 Search Order 这个 Tool, 回答用户的问题, 没有必要再让 Agent 思考一遍。

那我们就把代码再改造一下。

```

1 import re
2
3 answer_order_info = PromptTemplate(
4     template="请把下面的订单信息回复给用户: \n\n {order}?", input_variables=["order"]
5 )
6 answer_order_llm = LLMChain(llm=ChatOpenAI(temperature=0), prompt=answer_order_i
7
8 @tool("Search Order", return_direct=True)
9 def search_order(input:str)->str:
10     """useful for when you need to answer questions about customers orders"""
11     pattern = r"\d+[A-Z]+"
12     match = re.search(pattern, input)
13
14     order_number = input
15     if match:

```

```

16         order_number = match.group(0)
17     else:
18         return "请问您的订单号是多少? "
19     if order_number == ORDER_1:
20         return answer_order_llm.run(json.dumps(ORDER_1_DETAIL))
21     elif order_number == ORDER_2:
22         return answer_order_llm.run(json.dumps(ORDER_2_DETAIL))
23     else:
24         return f"对不起, 根据{input}没有找到您的订单"
25
26 from langchain.memory import ConversationBufferMemory
27 from langchain.chat_models import ChatOpenAI
28
29 tools = [search_order, recommend_product, faq]
30 chatllm=ChatOpenAI(temperature=0)
31 memory = ConversationBufferMemory(memory_key="chat_history", return_messages=True)
32 conversation_agent = initialize_agent(tools, chatllm,
33                                     agent="conversational-react-description",
34                                     memory=memory, verbose=True)

```

第一个改造还是在 Search Order 这个工具上的。首先，我们给这个 Tool 设置了一个参数，叫做 `return_direct = True`，这个参数是告诉 AI，在拿到这个工具的回复之后，不要再经过 Thought 那一步思考，直接把我们的回答给到用户就好了。设了这个参数之后，你就会发现 AI 不会在没有得到一个订单号的时候继续去反复思考，尝试使用工具，而是会直接去询问用户的订单号。

伴随着这个修改，对于查询到的订单号，我们就不能直接返回一个 JSON 字符串了，而是通过 `answer_order_llm` 这个工具来组织语言文字。

第二个改造是我们使用的 Agent，我们把 Agent 换成了 `converstional-react-description`。这样我们就支持多轮对话了，同时我们也把对应的 LLM 换成了 ChatOpenAI，这样成本更低。并且，我们还需要为这个 Agent 设置一下 memory。

改造好之后，我们不妨来看看，这个 AI 现在是不是终于智能一点了。

问题 1:


```
1 question1 = "我有一张订单，一直没有收到，能麻烦帮我查一下吗？"  
2 answer1 = conversation_agent.run(question1)  
3 print(answer1)
```

回答 1:

 复制代码

```
1 > Entering new AgentExecutor chain...  
2 Thought: Do I need to use a tool? Yes  
3 Action: Search Order  
4 Action Input: 我有一张订单，一直没有收到，能麻烦帮我查一下吗？  
5 Observation: 请问您的订单号是多少？  
6  
7 > Finished chain.  
8 请问您的订单号是多少？
```

问题 2:

 复制代码

```
1 question2 = "我的订单号是20230101ABC"  
2 answer2 = conversation_agent.run(question2)  
3 print(answer2)
```

回答 2:

 复制代码

```
1 > Entering new AgentExecutor chain...  
2 Thought: Do I need to use a tool? Yes  
3 Action: Search Order  
4 Action Input: 20230101ABC  
5 Observation:  
6 尊敬的用户，您的订单信息如下：  
7 订单号：20230101ABC  
8 订单状态：已发货  
9 发货日期：2023年1月3日  
10 预计送达日期：2023年1月5日  
11 如有任何疑问，请随时联系我们。感谢您的购买！  
12  
13 > Finished chain.
```


```
14
15 尊敬的用户，您的订单信息如下：
16 订单号：20230101ABC
17 订单状态：已发货
18 发货日期：2023年1月3日
19 预计送达日期：2023年1月5日
20 如有任何疑问，请随时联系我们。感谢您的购买！
```

### 问题 3:

 复制代码

```
1 question3 = "你们的退货政策是怎么样的？"
2 answer3 = conversation_agent.run(question3)
3 print(answer3)
```

### 回答 3:

 复制代码

```
1 > Entering new AgentExecutor chain...
2 Thought: Do I need to use a tool? Yes
3 Action: FAQ
4 Action Input: 退货政策
5 > Entering new VectorDBQA chain...
6 > Finished chain.
7 Observation: 自收到商品之日起7天内，如产品未使用、包装完好，您可以申请退货。某些特殊商品可能不
8 Thought:Do I need to use a tool? No
9 AI: Our return policy allows for returns within 7 days of receiving the product,
10 > Finished chain.
11 Our return policy allows for returns within 7 days of receiving the product, as l
```

可以看到 AI 能够在多轮对话里面，明白用户的意思，给出合理的答案。不过，最后一个问题它是用英文回答的，那怎么让它用中文来回答呢？这个问题就作为这一节课的思考题留给你啦！

好了，现在你已经有了一个有基本功能的电商客服聊天机器人了。你只需要在现有的这个代码上做一些改造，将自己的数据源导入进去，就可以拿真实的用户问题去试一试，看看效果怎么

样了。

## 小结

今天，我为你介绍了 Langchain 的 Agent 的基本功能。通过“先让 AI 做个选择题”的方式，Langchain 让 AI 自动为我们选择合适的 Tool 去调用。**我们可以把回答不同类型问题的 LLMChain 封装成不同的 Tool，也可以直接让 Tool 去调用内部查询订单状态的功能。**我也为你实际演示了将 Agent、Memory、VectorStore、LLMChain 组合在一起的过程，创建了一个有完整电商客服功能的聊天机器人。

我们对于 Langchain 的介绍也就告一段落了。作为大语言模型领域目前最火的一个开源项目，Langchain 有非常丰富的功能。我这里介绍的也只是它的核心功能，它支持丰富的 Tool、不同类型的 VectorStore 和内置的其他 LLMChain，都等待你自己去 [🔗 它的文档](#)里发掘了。

## 思考题

1. 在这一讲的最后，我们的例子里，AI 用英文回答了中文的 FAQ 问题，你能够尝试修改一下现在的代码，让 AI 用中文回复吗？
2. 上一讲里，我们介绍了 EntityMemory，但是在这一讲里我们并没有通过 EntityMemory 获取并查询订单信息。你能研究一下 Langchain 的文档，思考一下如果我们想要使用 EntityMemory 的话，应该怎么做吗？

欢迎你把思考后的结果分享到评论区，也欢迎你把这一讲分享给感兴趣的朋友，我们下一讲再见！

## 推荐阅读

Langchain 里面的 zero-shot-react-description 这个想法，来自一个知名的 AI 创业公司 AI21 Labs 的 [🔗 论文 MRKL Systems](#)。你有兴趣的话，可以去阅读一下。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

## 精选留言 (26)



孟健

2023-04-14 来自广东

最近的autogpt, agentgpt是不是都是这个思路

作者回复: 对, 其实都是和 ReAct 以及 MRKL 这两片论文的思路一脉相承的。其实看过点Langchain源码看那个基本上立刻就能想明白那个可以怎么做到。

共 3 条评论 >



6



Geek\_4ec46c

2023-04-14 来自福建

老师,这是我让让用中文回答的代码,你看下这样对不对? 或是说还有更加简单的方法?

```
```python
```

```
from langchain import LLMChain
```

```
from langchain.agents import Tool, AgentExecutor, ZeroShotAgent
```

```
import os
```

```
from langchain.chat_models import ChatOpenAI
```

```
def search_order(input: str) -> str:
```

```
    return "订单状态: 已发货; 发货日期: 2023-01-01; 预计送达时间: 2023-01-10"
```

```
def recommend_product(input: str) -> str:
```

```
    return "红色连衣裙"
```

```
def faq(intput: str) -> str:
```

```
    return "7天无理由退货"
```



```

tools = [
    Tool(
        name="Search Order", func=search_order,
        description="useful for when you need to answer questions about customers orders"
# 当您需要回答有关客户订单的问题时很有用
    ),
    Tool(name="Recommend Product", func=recommend_product,
        description="useful for when you need to answer questions about product recommen
dations" # 你需要回答关于产品推荐的问题时很有用
    ),
    Tool(name="FAQ", func=faq,
        description="useful for when you need to answer questions about shopping policies, li
ke return policy, shipping policy, etc."
        # 当您需要回答关于购物政策的问题时，例如退货政策、运输政策等，这将非常有用。
    )
]

```

PREFIX = ""Answer the following questions as best you can. You have access to the followi  
ng tools: (所有的回答都用中文返回)""

```

prompt_ = ZeroShotAgent.create_prompt(
    tools,
    prefix=PREFIX,
    input_variables=["input", "agent_scratchpad"]
)

```

```

llm_chain = LLMChain(llm=ChatOpenAI(temperature=0), prompt=prompt_)
agent = ZeroShotAgent(llm_chain=llm_chain, tools=tools, verbose=True)

```

```

agent_chain = AgentExecutor.from_agent_and_tools(agent=agent, tools=tools, verbose=Tr
ue)
result = agent_chain.run("我有一张订单，订单号是 2022ABCDE，一直没有收到，能麻烦帮
我查一下吗? ")
print(result)
```

```

作者回复: 👍，我其实就是想要大家

1. 理解这件事情是通过Prompt来解决

2. 去看一下Langchain的文档，看看怎么根据自己的需要修改Prompt



👍 3



**Leo**

2023-05-04 来自浙江

老师，我想基于这个做一个对人的检索，用户输入姓名、年龄、性别等能找到相关的人，这个用embedding index能实现吗？或者有其他思路可以实现吗？求解答

作者回复: 1. 这个肯定不适合用embedding，简单的数据库SQL就可以了

2. 这个应用涉及到PII这样的个人隐私信息，不要做，容易违法



👍 1



**李蕾**

2023-04-19 来自广东

关于老师思考题的第一个问题，经过自己的尝试，比较简单的方式是在Template中明确要求必须用中文返回即可。

```
answer_order_info = PromptTemplate( template="请把下面的订单信息用中文回复给用户： \n\n {order}?", input_variables=["order"])
```

我的第一想法和Geek\_4ec46c一样，也是尝试着在Prompt中规定返回必须是中文，但是没有生效，可能是自己当前的功力不够导致的bug，哈哈

作者回复: 可以去看一下对应文档，通过 ai\_prefix 的参数来操作，而不需要整个PromptTemplate重写。



👍 1



**树静风止**

2023-04-17 来自北京

这么强大超期的课程，人怎么这么少呢

作者回复: 欢迎宣传转发给你的朋友们 😊



👍 1



**智能**

2023-04-14 来自北京

所以这里chatGPT其实相当于一个调度者？识别用户意图然后调用其他应用

作者回复: 其实比调度者更加“高级”，你可以认为我们其实是通过提示语，让ChatGPT模拟人思考问题的过程



👍 1



**Jacob.C**

2023-04-14 来自广东

老师，请问 agent 的一个 tool如果需要 llm 做两件事再回来，应该怎么玩呢？

作者回复: 那就用一个前两讲介绍过的 SequentialChain 作为LLMChain呀。Tool封装一下这个SequentialChain的LLMChain就好了。



👍 1



**Jacob.C**

2023-04-14 来自广东

解决的方法也不复杂，我们只需要调整一下 search\_order 这个 Tool 的提示语。通过这个提示语，Agent 会知道，这个工具就应该在找不到订单的时候，告诉用户找不到订单或者请它再次确认。这个时候，它会根据这个答案去回复用户。下面是对应修改运行后的结果。

老师这里两段代码实在看不出来修改了啥，麻烦看出来的人说明一下！

作者回复: 有个地方代码贴错了，编辑稍后会帮忙修正。Prompt没有更正之前是

```
@tool("Search Order")
def search_order(input:str)->str:
    """useful for when you need to answer questions about customers orders"""
    if input.strip() == ORDER_1:
        return json.dumps(ORDER_1_DETAIL)
    elif input.strip() == ORDER_2:
        return json.dumps(ORDER_2_DETAIL)
    else:
        return f"对不起，根据{input}没有找到您的订单"
```



1



**Yezhiwei**

2023-04-14 来自北京

谢谢老师哈，这篇文章牛皮plus



1



**~鹏~**

2023-05-18 来自北京

请问下我用chatglm我试了下上面写的agent例子，发现不支持，这个只能openai可以用吗

作者回复: ChatGLM的推理能力如果是 6B 的版本的话，目前还没有那么强。

对于需要相对复杂推理能力的，目前看起来还是需要 openai 或者在 llama 比较大的模型上微调才能有一定效果



**风**

2023-04-30 来自北京

你好，我尝试使用ChatGLM实现这一节的内容，但是效果很差，经常没办法像OpenAI那样进行“思考”，最后返回正常的结果。有什么好的办法吗？或者，有什么其他可用的中文LLM模型可以使用agent和tool实现类似的功能。

作者回复: “思考”能力现在其他模型和OpenAI差距很大，而且GPT-3.5和GPT-4差距都不小，最好能用GPT-4



**江湖中人**

2023-04-21 来自浙江

老师，请教一下，如果想做一个特定行业的客服机器人，有一些私有的数据需要喂给AI，是有现成的东西可以用，还是利用我们课程学的东西去开发训练，可以大致梳理一下流程吗？很多有这种需求的但是对技术并不了解的人，他们要怎么做呢？

作者回复: 第17和18讲的内容就是为了解决这个问题的？第17讲对于客服问题的解法是否满足你的需求呢？

共 2 条评论 >



**Geek\_3d7708**

2023-04-18 来自浙江

- 1、 LLM 换成了 ChatOpenAI，不知道怎么 回答过程都是英文了；
- 2、converstional-react-description 采用这个，回答就不是从知识库了。比如我用藤野显示，问鲁迅老师是谁？zero-shot-react-description 的时候，是藤野先生，但是换成converstional-react-description 就变成 陈寅格，不知道哪来的。

作者回复: 1. 这个和Prompt有关，Langchain默认的Prompt是英文，会导致输出结果变成英文。这个你可以通过 ai\_prefix 之类的参数，加入一段要求输出是中文的文本就好了。

2. conversational-react-description 你需要指定对应的Tools是VBQA并且tools的description要能够让AI理解这个问题能够靠这个Tool来解决。



**xianbin.yang**

2023-04-18 来自浙江

老师，很喜欢您这个专栏，您除了这个专栏，还有其他的博客、社群或者知识星球吗？

作者回复: 有一个和朋友一起新做的微信公众号“AI炼金术”，不过不保证长期更新.....



**Geek\_3d7708**

2023-04-18 来自中国香港

EntityMemory 每个不同的客户，保存的数据会不会冲突？比如A的数据，B也访问到了？

作者回复: 需要你为自己为每一个客户单独维护Session，每个用户有一个自己的memory呀。



**Geek\_4ec46c**

2023-04-16 来自中国台湾

解决的方法也不复杂，我们只需要调整一下 search\_order 这个 Tool 的提示语。

在这个位置的代码和上面的代码都是一样的,好像没有任何调整.....（老师是你没改好？）

作者回复: 编辑提醒我了, 稿件里我粘贴代码的时候贴错了上面一段, 稍后会修正过来。



**张弛**

2023-04-15 来自中国台湾

```
question3 = "你们的退货政策是怎么样的?"
"answer3 = conversation_agent.run(question3)
print(answer2)
```

这段代码的第三行, print的参数错了, 应该是answer3

作者回复: 感谢提醒, 我请编辑修正一下。



**张弛**

2023-04-15 来自中国台湾

```
question1 = "我有一张订单, 一直没有收到, 能麻烦帮我查一下吗?"
answer1 = conversation_agent.run(question)
print(answer1)
```

这段代码的第二行的参数填错了, 应该是question1

作者回复: 感谢提醒, 编辑提醒我了, 稍后会修正一下。



**Toni**

2023-04-15 来自瑞士

对思考题的思考:

经多轮对话后(例子中的第三轮对话), AI 用英文回答了中文的 FAQ 问题, 你能够尝试修改一下现在的代码, 让 AI 用中文回复吗?

连续的三个问题:

```
question1 = "我有一张订单, 一直没有收到, 能麻烦帮我查一下吗?"
answer1 = conversation_agent.run(question1)
print(answer1)
```

```
question2 = "我的订单号是20230101ABC"  
answer2 = conversation_agent.run(question2)  
print(answer2)
```

```
question3 = "你们的退货政策是怎么样的? "  
answer3 = conversation_agent.run(question3)  
print(answer3)
```

AI 的回答:

> Entering new AgentExecutor chain...

Thought: Do I need to use a tool? Yes

Action: Search Order

Action Input: 订单号码

Observation: 请问您的订单号是多少?

> Finished chain.

请问您的订单号是多少?

> Entering new AgentExecutor chain...

Thought: Do I need to use a tool? Yes

Action: Search Order

Action Input: 20230101ABC

Observation: 尊敬的用户，感谢您选择我们的服务。以下是您的订单信息：

订单号码：20230101ABC

订单状态：已发货

发货日期：2023年1月3日

预计送达日期：2023年1月5日

如果您有任何疑问或需要进一步的帮助，请随时联系我们。祝您购物愉快！

> Finished chain.

尊敬的用户，感谢您选择我们的服务。以下是您的订单信息：

订单号码：20230101ABC

订单状态：已发货

发货日期：2023年1月3日

预计送达日期：2023年1月5日

如果您有任何疑问或需要进一步的帮助，请随时联系我们。祝您购物愉快！

> Entering new AgentExecutor chain...



Thought: Do I need to use a tool? Yes

Action: FAQ

Action Input: "退货政策"

> Entering new VectorDBQA chain...

> Finished chain.

Observation: 自收到商品之日起7天内，如产品未使用、包装完好，您可以申请退货。某些特殊商品可能不支持退货，请在购买前查看商品详情页面的退货政策。

> Finished chain.

自收到商品之日起7天内，如产品未使用、包装完好，您可以申请退货。某些特殊商品可能不支持退货，请在购买前查看商品详情页面的退货政策。



**张弛**

2023-04-15 来自中国台湾

老师您好！本次案例中的ecommerce\_products.csv文件好像并未上传到github，麻烦补充，谢谢！

作者回复: 刚刚上传了，感谢提醒

