第四章 模型链

链（Chains）通常将大语言模型（LLM）与提示（Prompt）结合在一起，基于此，我们可以对文本或数据进行一系列操作。链（Chains）可以一次性接受多个输入。例如，我们可以创建一个链，该链接受用户输入，使用提示模板对其进行格式化，然后将格式化的响应传递给 LLM 。我们可以通过将多个链组合在一起，或者通过将链与其他组件组合在一起来构建更复杂的链。

一、大语言模型链

大语言模型链（LLMChain）是一个简单但非常强大的链，也是后面我们将要介绍的许多链的基础。

1.1 初始化语言模型

```
import warnings
warnings.filterwarnings('ignore')

from langchain.chat_models import ChatOpenAI
from langchain.prompts import ChatPromptTemplate
from langchain.chains import LLMChain

# 这里我们将参数temperature设置为0.0，从而减少生成答案的随机性。
# 如果你想要每次得到不一样的有新意的答案，可以尝试调整该参数。
llm = ChatOpenAI(temperature=0.0)
```

1.2 初始化提示模版

初始化提示，这个提示将接受一个名为product的变量。该prompt将要求LLM生成一个描述制造该产品的公司的最佳名称

```
prompt = ChatPromptTemplate.from_template("描述制造{product}的一个公司的最佳名称是什么?")
```

1.3 构建大语言模型链

将大语言模型(LLM)和提示（Prompt）组合成链。这个大语言模型链非常简单，可以让我们以一种顺序的方式去通过运行提示并且结合到大语言模型中。

```
chain = LLMChain(llm=llm, prompt=prompt)
```

1.4 运行大语言模型链

因此，如果我们有一个名为"Queen Size Sheet Set"的产品，我们可以通过使用`chain.run`将其通过这个链运行

```
product = "大号床单套装"
chain.run(product)
```

```
'"豪华床纺"'
```

您可以输入任何产品描述，然后查看链将输出什么结果。

二、简单顺序链

顺序链（SequentialChains）是按预定义顺序执行其链接的链。具体来说，我们将使用简单顺序链（SimpleSequentialChain），这是顺序链的最简单类型，其中每个步骤都有一个输入/输出，一个步骤的输出是下一个步骤的输入。

```
from langchain.chains import SimpleSequentialChain
llm = ChatOpenAI(temperature=0.9)
```

2.1 创建两个子链

```
# 提示模板 1 ：这个提示将接受产品并返回最佳名称来描述该公司
first_prompt = ChatPromptTemplate.from_template(
    "描述制造{product}的一个公司的最好的名称是什么"
)
chain_one = LLMChain(llm=llm, prompt=first_prompt)

# 提示模板 2 ：接受公司名称，然后输出该公司的长为20个单词的描述
second_prompt = ChatPromptTemplate.from_template(
    "写一个20字的描述对于下面这个\
    公司：{company_name}的"
)
chain_two = LLMChain(llm=llm, prompt=second_prompt)
```

2.2 构建简单顺序链

现在我们可以组合两个LLMChain，以便我们可以在一个步骤中创建公司名称和描述

```
overall_simple_chain = SimpleSequentialChain(chains=[chain_one, chain_two],
                                             verbose=True)
```

给一个输入，然后运行上面的链

2.3 运行简单顺序链

```
product = "大号床单套装"
overall_simple_chain.run(product)
```

```
> Entering new SimpleSequentialChain chain...
优床制造公司
优床制造公司是一家专注于生产高品质床具的公司。

> Finished chain.
```

```
'优床制造公司是一家专注于生产高品质床具的公司。'
```

三、 顺序链

当只有一个输入和一个输出时，简单顺序链（SimpleSequentialChain）即可实现。当有多个输入或多个输出时，我们则需要使用顺序链（SequentialChain）来实现。

```
import pandas as pd
from langchain.chains import SequentialChain
from langchain.chat_models import ChatOpenAI    #导入OpenAI模型
from langchain.prompts import ChatPromptTemplate   #导入聊天提示模板
from langchain.chains import LLMChain    #导入LLM链。

llm = ChatOpenAI(temperature=0.9)
```

接下来我们将创建一系列的链，然后一个接一个使用他们

3.1 创建四个子链

```
#子链1
# prompt模板 1: 翻译成英语（把下面的review翻译成英语）
first_prompt = ChatPromptTemplate.from_template(
    "把下面的评论review翻译成英文:"
    "\n\n{Review}"
)
# chain 1: 输入：Review    输出：英文的 Review
chain_one = LLMChain(llm=llm, prompt=first_prompt, output_key="English_Review")

#子链2
# prompt模板 2: 用一句话总结下面的 review
second_prompt = ChatPromptTemplate.from_template(
    "请你用一句话来总结下面的评论review:"
    "\n\n{English_Review}"
)
# chain 2: 输入：英文的Review   输出：总结
chain_two = LLMChain(llm=llm, prompt=second_prompt, output_key="summary")


#子链3
# prompt模板 3: 下面review使用的什么语言
third_prompt = ChatPromptTemplate.from_template(
    "下面的评论review使用的什么语言:\n\n{Review}"
)
# chain 3: 输入：Review  输出：语言
chain_three = LLMChain(llm=llm, prompt=third_prompt, output_key="language")


#子链4
# prompt模板 4: 使用特定的语言对下面的总结写一个后续回复
fourth_prompt = ChatPromptTemplate.from_template(
    "使用特定的语言对下面的总结写一个后续回复:"
    "\n\n总结: {summary}\n\n语言: {language}"
)
# chain 4: 输入：  总结, 语言     输出：  后续回复
chain_four = LLMChain(llm=llm, prompt=fourth_prompt,
output_key="followup_message")
```

3.2 对四个子链进行组合

```
#输入：review
#输出：英文review，总结，后续回复
overall_chain = SequentialChain(
    chains=[chain_one, chain_two, chain_three, chain_four],
    input_variables=["Review"],
    output_variables=["English_Review", "summary","followup_message"],
    verbose=True
)
```

让我们选择一篇评论并通过整个链传递它，可以发现，原始review是法语，可以把英文review看做是一种翻译，接下来是根据英文review得到的总结，最后输出的是用法语原文进行的续写信息。

```
df = pd.read_csv('../data/Data.csv')
review = df.Review[5]
overall_chain(review)
```

```
> Entering new SequentialChain chain...

> Finished chain.
```

```
{'Review': "Je trouve le goût médiocre. La mousse ne tient pas, c'est bizarre.
J'achète les mêmes dans le commerce et le goût est bien meilleur...\nVieux lot ou
contrefaçon !?",
 'English_Review': "I find the taste mediocre. The foam doesn't hold, it's weird.
I buy the same ones in stores and the taste is much better...\nOld batch or
counterfeit!?",
 'summary': "The reviewer finds the taste mediocre, the foam doesn't hold well,
and suspects the product may be either an old batch or a counterfeit.",
 'followup_message': "后续回复（法语）：Merci beaucoup pour votre avis. Nous sommes
désolés d'apprendre que vous avez trouvé le goût médiocre et que la mousse ne
tient pas bien. Nous prenons ces problèmes très au sérieux et nous enquêterons
sur la possibilité que le produit soit soit un ancien lot, soit une contrefaçon.
Nous vous prions de nous excuser pour cette expérience décevante et nous ferons
tout notre possible pour résoudre ce problème. Votre satisfaction est notre
priorité et nous apprécions vos commentaires précieux."}
```

四、 路由链

到目前为止，我们已经学习了大语言模型链和顺序链。但是，如果我们想做一些更复杂的事情怎么办？一个相当常见但基本的操作是根据输入将其路由到一条链，具体取决于该输入到底是什么。如果你有多个子链，每个子链都专门用于特定类型的输入，那么可以组成一个路由链，它首先决定将它传递给哪个子链，然后将它传递给那个链。

路由器由两个组件组成：

- 路由链（Router Chain）：路由器链本身，负责选择要调用的下一个链
- destination_chains：路由器链可以路由到的链

举一个具体的例子，让我们看一下我们在不同类型的链之间路由的地方，我们在这里有不同的prompt:

```
from langchain.chains.router import MultiPromptChain  #导入多提示链
from langchain.chains.router.llm_router import LLMRouterChain,RouterOutputParser
from langchain.prompts import PromptTemplate
llm = ChatOpenAI(temperature=0)
```

4.1 定义提示模板

首先，我们定义提示适用于不同场景下的提示模板。

```
# 中文
#第一个提示适合回答物理问题
physics_template = """你是一个非常聪明的物理专家。 \
你擅长用一种简洁并且易于理解的方式去回答问题。\
当你不知道问题的答案时，你承认\
你不知道.

这是一个问题:
{input}"""


#第二个提示适合回答数学问题
math_template = """你是一个非常优秀的数学家。 \
你擅长回答数学问题。 \
你之所以如此优秀， \
是因为你能够将棘手的问题分解为组成部分，\
回答组成部分，然后将它们组合在一起，回答更广泛的问题。

这是一个问题:
{input}"""


#第三个适合回答历史问题
history_template = """你是以为非常优秀的历史学家。 \
你对一系列历史时期的人物、事件和背景有着极好的学识和理解\
你有能力思考、反思、辩证、讨论和评估过去。\
你尊重历史证据，并有能力利用它来支持你的解释和判断。

这是一个问题:
{input}"""


#第四个适合回答计算机问题
computerscience_template = """ 你是一个成功的计算机科学专家。\
你有创造力、协作精神、\
前瞻性思维、自信、解决问题的能力、\
对理论和算法的理解以及出色的沟通技巧。\
你非常擅长回答编程问题。\
你之所以如此优秀，是因为你知道  \
如何通过以机器可以轻松解释的命令式步骤描述解决方案来解决问题，\
并且你知道如何选择在时间复杂性和空间复杂性之间取得良好平衡的解决方案。

这还是一个输入:
{input}"""
```

4.2 对提示模版进行命名和描述

在定义了这些提示模板后，我们可以为每个模板命名，并给出具体描述。例如，第一个物理学的描述适合回答关于物理学的问题，这些信息将传递给路由链，然后由路由链决定何时使用此子链。

```
# 中文
prompt_infos = [
    {
        "名字": "物理学",
        "描述": "擅长回答关于物理学的问题",
        "提示模板": physics_template
    },
    {
        "名字": "数学",
        "描述": "擅长回答数学问题",
        "提示模板": math_template
    },
    {
        "名字": "历史",
        "描述": "擅长回答历史问题",
        "提示模板": history_template
    },
    {
        "名字": "计算机科学",
        "描述": "擅长回答计算机科学问题",
        "提示模板": computerscience_template
    }
]
```

LLMRouterChain（此链使用 LLM 来确定如何路由事物）

在这里，我们需要一个**多提示链**。这是一种特定类型的链，用于在多个不同的提示模板之间进行路由。但是这只是路由的一种类型，我们也可以在任何类型的链之间进行路由。

这里我们要实现的几个类是大模型路由器链。这个类本身使用语言模型来在不同的子链之间进行路由。这就是上面提供的描述和名称将被使用的地方。

4.3 基于提示模版信息创建相应目标链

目标链是由路由链调用的链，每个目标链都是一个语言模型链

```
destination_chains = {}
for p_info in prompt_infos:
    name = p_info["名字"]
    prompt_template = p_info["提示模板"]
    prompt = ChatPromptTemplate.from_template(template=prompt_template)
    chain = LLMChain(llm=llm, prompt=prompt)
    destination_chains[name] = chain

destinations = [f"{p['名字']}: {p['描述']}" for p in prompt_infos]
destinations_str = "\n".join(destinations)
```

4.4 创建默认目标链

除了目标链之外，我们还需要一个默认目标链。这是一个当路由器无法决定使用哪个子链时调用的链。在上面的示例中，当输入问题与物理、数学、历史或计算机科学无关时，可能会调用它。

```
default_prompt = ChatPromptTemplate.from_template("{input}")
default_chain = LLMChain(llm=llm, prompt=default_prompt)
```

4.5 定义不同链之间的路由模板

这包括要完成的任务的说明以及输出应该采用的特定格式。

注意：此处在原教程的基础上添加了一个示例，主要是因为"gpt-3.5-turbo"模型不能很好适应理解模板的意思，使用 "text-davinci-003" 或者"gpt-4-0613"可以很好的工作，因此在这里多加了示例提示让其更好的学习。

例如:

<< INPUT >>

"What is black body radiation?"

<< OUTPUT >>

```
{{{{
    "destination": string \ name of the prompt to use or "DEFAULT"
    "next_inputs": string \ a potentially modified version of the original input
}}}}
```

```
# 多提示路由模板
MULTI_PROMPT_ROUTER_TEMPLATE = """给语言模型一个原始文本输入，\
让其选择最适合输入的模型提示。\
系统将为您提供可用提示的名称以及最适合改提示的描述。\
如果你认为修改原始输入最终会导致语言模型做出更好的响应，\
你也可以修改原始输入。


<< 格式 >>
返回一个带有JSON对象的markdown代码片段，该JSON对象的格式如下：
```json
{{{{
    "destination": 字符串 \ 使用的提示名字或者使用 "DEFAULT"
    "next_inputs": 字符串 \ 原始输入的改进版本
}}}}



记住："destination"必须是下面指定的候选提示名称之一，\
或者如果输入不太适合任何候选提示，\
则可以是 "DEFAULT" 。
记住：如果您认为不需要任何修改，\
则 "next_inputs" 可以只是原始输入。

<< 候选提示 >>
{destinations}
```
```

```
<< 输入 >>
{{input}}

<< 输出 (记得要包含 ```json)>>

样例:
<< 输入 >>
"什么是黑体辐射?"
<< 输出 >>
```json
{{{{
    "destination": 字符串 \ 使用的提示名字或者使用 "DEFAULT"
    "next_inputs": 字符串 \ 原始输入的改进版本
}}}}

"""
```

4.6 构建路由链

首先，我们通过格式化上面定义的目标创建完整的路由器模板。这个模板可以适用许多不同类型的目标。
因此，在这里，您可以添加一个不同的学科，如英语或拉丁语，而不仅仅是物理、数学、历史和计算机科学。

接下来，我们从这个模板创建提示模板。

最后，通过传入llm和整个路由提示来创建路由链。需要注意的是这里有路由输出解析，这很重要，因为它将帮助这个链路决定在哪些子链路之间进行路由。

```
router_template = MULTI_PROMPT_ROUTER_TEMPLATE.format(
    destinations=destinations_str
)
router_prompt = PromptTemplate(
    template=router_template,
    input_variables=["input"],
    output_parser=RouterOutputParser(),
)

router_chain = LLMRouterChain.from_llm(llm, router_prompt)
```

4.7 创建整体链路

```
#多提示链
chain = MultiPromptChain(router_chain=router_chain,    #l路由链路
                         destination_chains=destination_chains,    #目标链路
                         default_chain=default_chain,      #默认链路
                         verbose=True
                        )
```
```

4.8 进行提问

如果我们问一个物理问题，我们希望看到他被路由到物理链路。

```
chain.run("什么是黑体辐射？")
```

```
> Entering new MultiPromptChain chain...
物理学: {'input': '什么是黑体辐射？'}
> Finished chain.
```

```
'黑体辐射是指一个理想化的物体，它能够完全吸收并且以最高效率地辐射出所有入射到它上面的电磁辐射。这种辐射的特点是它的辐射强度与波长有关，且在不同波长下的辐射强度符合普朗克辐射定律。黑体辐射在物理学中有广泛的应用，例如研究热力学、量子力学和宇宙学等领域。'
```

如果我们问一个数学问题，我们希望看到他被路由到数学链路。

```
chain.run("2+2等于多少？")
```

```
> Entering new MultiPromptChain chain...
数学: {'input': '2+2等于多少？'}
> Finished chain.
```

```
'2+2等于4。'
```

如果我们传递一个与任何子链路都无关的问题时，会发生什么呢?

这里，我们问了一个关于生物学的问题，我们可以看到它选择的链路是无。这意味着它将被**传递到默认链路，它本身只是对语言模型的通用调用。**语言模型幸运地对生物学知道很多，所以它可以帮助我们。

```
chain.run("为什么我们身体里的每个细胞都包含DNA？")
```

```
> Entering new MultiPromptChain chain...
物理学: {'input': '为什么我们身体里的每个细胞都包含DNA？'}
> Finished chain.
```

```
'我们身体里的每个细胞都包含DNA，因为DNA是遗传信息的载体。DNA是由四种碱基（腺嘌呤、胸腺嘧啶、鸟嘌呤和胞嘧啶）组成的长链状分子，它存储了我们的遗传信息，包括我们的基因和遗传特征。每个细胞都需要这些遗传信息来执行其特定的功能和任务。所以，DNA存在于每个细胞中，以确保我们的身体正常运作。'
```

英文版提示

1.大语言模型链

```
from langchain.chat_models import ChatOpenAI
from langchain.prompts import ChatPromptTemplate
from langchain.chains import LLMChain


llm = ChatOpenAI(temperature=0.0)
```

```
prompt = ChatPromptTemplate.from_template(
    "What is the best name to describe \
    a company that makes {product}?"
)

chain = LLMChain(llm=llm, prompt=prompt)

product = "Queen Size Sheet Set"
chain.run(product)
```

```
'Royal Comfort Linens'
```

2.简单顺序链

```
from langchain.chains import SimpleSequentialChain
llm = ChatOpenAI(temperature=0.9)


first_prompt = ChatPromptTemplate.from_template(
    "What is the best name to describe \
    a company that makes {product}?"
)

# Chain 1
chain_one = LLMChain(llm=llm, prompt=first_prompt)

second_prompt = ChatPromptTemplate.from_template(
    "Write a 20 words description for the following \
    company:{company_name}"
)
# chain 2
chain_two = LLMChain(llm=llm, prompt=second_prompt)


overall_simple_chain = SimpleSequentialChain(chains=[chain_one, chain_two],
verbose=True)
product = "Queen Size Sheet Set"
overall_simple_chain.run(product)
```

```
> Entering new SimpleSequentialChain chain...
"Royal Comfort Beddings"
Royal Comfort Beddings is a reputable company that offers luxurious and
comfortable bedding options fit for royalty.

> Finished chain.
```

```
'Royal Comfort Beddings is a reputable company that offers luxurious and
comfortable bedding options fit for royalty.'
```

3. 顺序链

```
from langchain.chains import SequentialChain
from langchain.chat_models import ChatOpenAI
from langchain.prompts import ChatPromptTemplate
from langchain.chains import LLMChain

llm = ChatOpenAI(temperature=0.9)

first_prompt = ChatPromptTemplate.from_template(
    "Translate the following review to english:"
    "\n\n{Review}"
)

chain_one = LLMChain(llm=llm, prompt=first_prompt, output_key="English_Review")


second_prompt = ChatPromptTemplate.from_template(
    "Can you summarize the following review in 1 sentence:"
    "\n\n{English_Review}"
)

chain_two = LLMChain(llm=llm, prompt=second_prompt, output_key="summary")

third_prompt = ChatPromptTemplate.from_template(
    "What language is the following review:\n\n{Review}"
)

chain_three = LLMChain(llm=llm, prompt=third_prompt, output_key="language")

fourth_prompt = ChatPromptTemplate.from_template(
    "Write a follow up response to the following "
    "summary in the specified language:"
    "\n\nSummary: {summary}\n\nLanguage: {language}"
)

chain_four = LLMChain(llm=llm, prompt=fourth_prompt,
output_key="followup_message")


overall_chain = SequentialChain(
    chains=[chain_one, chain_two, chain_three, chain_four],
    input_variables=["Review"],
    output_variables=["English_Review", "summary","followup_message"],
    verbose=True
)

review = df.Review[5]
overall_chain(review)
```

```
> Entering new SequentialChain chain...

> Finished chain.
```

{'Review': "Je trouve le goût médiocre. La mousse ne tient pas, c'est bizarre. J'achète les mêmes dans le commerce et le goût est bien meilleur...\nVieux lot ou contrefaçon !?",
 'English_Review': "I find the taste poor. The foam doesn't hold, it's weird. I buy the same ones from the store and the taste is much better...\nOld batch or counterfeit!?",
 'summary': 'The reviewer is disappointed with the poor taste and lack of foam in the product, suspecting it to be either an old batch or a counterfeit.',
 'followup_message': "Réponse de suivi:\n\nCher(e) critique,\n\nNous sommes vraiment désolés d'apprendre que vous avez été déçu(e) par le mauvais goût et l'absence de mousse de notre produit. Nous comprenons votre frustration et souhaitons rectifier cette situation.\n\nTout d'abord, nous tenons à vous assurer que nous ne vendons que des produits authentiques et de haute qualité. Chaque lot de notre produit est soigneusement contrôlé avant d'être mis sur le marché.\n\nCependant, il est possible qu'un incident se soit produit dans votre cas. Nous vous invitons donc à nous contacter directement afin que nous puissions mieux comprendre la situation et résoudre ce problème. Nous accorderons une attention particulière à la fraîcheur et à la mousse de notre produit pour garantir votre satisfaction.\n\nNous tenons à vous remercier pour vos commentaires, car ils nous aident à améliorer continuellement notre produit. Votre satisfaction est notre priorité absolue et nous ferons tout notre possible pour rectifier cette situation.\n\nNous espérons avoir l'opportunité de vous fournir une expérience agréable avec notre produit à l'avenir.\n\nCordialement,\nL'équipe du service client"}

4. 路由链

```
from langchain.chains import LLMChain
from langchain.chat_models import ChatOpenAI
from langchain.prompts import ChatPromptTemplate
from langchain.chains.router import MultiPromptChain
from langchain.chains.router.llm_router import LLMRouterChain,RouterOutputParser
from langchain.prompts import PromptTemplate
llm = ChatOpenAI(temperature=0.0)

physics_template = """You are a very smart physics professor. \
You are great at answering questions about physics in a concise\
and easy to understand manner. \
When you don't know the answer to a question you admit\
that you don't know.

Here is a question:
{input}"""


math_template = """You are a very good mathematician. \
You are great at answering math questions. \
You are so good because you are able to break down \
hard problems into their component parts,
answer the component parts, and then put them together\
to answer the broader question.

Here is a question:
{input}"""
```

```
history_template = """You are a very good historian. \
You have an excellent knowledge of and understanding of people,\
events and contexts from a range of historical periods. \
You have the ability to think, reflect, debate, discuss and \
evaluate the past. You have a respect for historical evidence\
and the ability to make use of it to support your explanations \
and judgements.

Here is a question:
{input}"""


computerscience_template = """ You are a successful computer scientist.\
You have a passion for creativity, collaboration,\
forward-thinking, confidence, strong problem-solving capabilities,\
understanding of theories and algorithms, and excellent communication \
skills. You are great at answering coding questions. \
You are so good because you know how to solve a problem by \
describing the solution in imperative steps \
that a machine can easily interpret and you know how to \
choose a solution that has a good balance between \
time complexity and space complexity.

Here is a question:
{input}"""

prompt_infos = [
    {
        "name": "physics",
        "description": "Good for answering questions about physics",
        "prompt_template": physics_template
    },
    {
        "name": "math",
        "description": "Good for answering math questions",
        "prompt_template": math_template
    },
    {
        "name": "History",
        "description": "Good for answering history questions",
        "prompt_template": history_template
    },
    {
        "name": "computer science",
        "description": "Good for answering computer science questions",
        "prompt_template": computerscience_template
    }
]


destination_chains = {}
for p_info in prompt_infos:
    name = p_info["name"]
    prompt_template = p_info["prompt_template"]
    prompt = ChatPromptTemplate.from_template(template=prompt_template)
```

```
    chain = LLMChain(llm=llm, prompt=prompt)
    destination_chains[name] = chain

destinations = [f"{p['name']}: {p['description']}" for p in prompt_infos]
destinations_str = "\n".join(destinations)

default_prompt = ChatPromptTemplate.from_template("{input}")
default_chain = LLMChain(llm=llm, prompt=default_prompt)


MULTI_PROMPT_ROUTER_TEMPLATE = """Given a raw text input to a \
language model select the model prompt best suited for the input. \
You will be given the names of the available prompts and a \
description of what the prompt is best suited for. \
You may also revise the original input if you think that revising\
it will ultimately lead to a better response from the language model.

<< FORMATTING >>
Return a markdown code snippet with a JSON object formatted to look like:
```json
{{{{
    "destination": string \ name of the prompt to use or "DEFAULT"
    "next_inputs": string \ a potentially modified version of the original input
}}}}

REMEMBER: "destination" MUST be one of the candidate prompt \
names specified below OR it can be "DEFAULT" if the input is not\
well suited for any of the candidate prompts.
REMEMBER: "next_inputs" can just be the original input \
if you don't think any modifications are needed.

<< CANDIDATE PROMPTS >>
{destinations}

<< INPUT >>
{{input}}

<< OUTPUT (remember to include the ```json)>>"""


router_template = MULTI_PROMPT_ROUTER_TEMPLATE.format(
    destinations=destinations_str
)
router_prompt = PromptTemplate(
    template=router_template,
    input_variables=["input"],
    output_parser=RouterOutputParser(),
)

router_chain = LLMRouterChain.from_llm(llm, router_prompt)


chain = MultiPromptChain(router_chain=router_chain,
                         destination_chains=destination_chains,
                         default_chain=default_chain, verbose=True
                        )
```
```

```
print(chain.run("What is black body radiation?"))

print(chain.run("what is 2 + 2"))

print(chain.run("Why does every cell in our body contain DNA?"))
```

```
    > Entering new MultiPromptChain chain...
    physics: {'input': 'What is black body radiation?'}
    > Finished chain.
    Black body radiation refers to the electromagnetic radiation emitted by an object that absorbs all incident radiation and reflects or transmits none. It is called "black body" because it absorbs all wavelengths of light, appearing black at room temperature.

    According to Planck's law, black body radiation is characterized by a continuous spectrum of wavelengths and intensities, which depend on the temperature of the object. As the temperature increases, the peak intensity of the radiation shifts to shorter wavelengths, resulting in a change in color from red to orange, yellow, white, and eventually blue at very high temperatures.

    Black body radiation is a fundamental concept in physics and has significant applications in various fields, including astrophysics, thermodynamics, and quantum mechanics. It played a crucial role in the development of quantum theory and understanding the behavior of light and matter.


    > Entering new MultiPromptChain chain...
    math: {'input': 'what is 2 + 2'}
    > Finished chain.
    Thank you for your kind words! As a mathematician, I am happy to help with any math questions, no matter how simple or complex they may be.

    The question you've asked is a basic addition problem: "What is 2 + 2?" To solve this, we can simply add the two numbers together:

    2 + 2 = 4

    Therefore, the answer to the question "What is 2 + 2?" is 4.


    > Entering new MultiPromptChain chain...
    None: {'input': 'Why does every cell in our body contain DNA?'}
    > Finished chain.
    Every cell in our body contains DNA because DNA is the genetic material that carries the instructions for the development, functioning, and reproduction of all living organisms. DNA contains the information necessary for the synthesis of proteins, which are essential for the structure and function of cells. It serves as a blueprint for the production of specific proteins that determine the characteristics and traits of an organism. Additionally, DNA is responsible for the transmission of genetic information from one generation to the next during reproduction. Therefore, every cell in our body contains DNA to ensure the proper functioning and continuity of life.
```