Dynamically route logic based on input

This notebook covers how to do routing in the LangChain Expression Language.

Routing allows you to create non-deterministic chains where the output of a previous step defines the next step. Routing helps provide structure and consistency around interactions with LLMs.

There are two ways to perform routing:

1. Using a `RunnableBranch`.
2. Writing custom factory function that takes the input of a previous step and returns a **runnable**. Importantly, this should return a **runnable** and NOT actually execute.

We'll illustrate both methods using a two step sequence where the first step classifies an input question as being about `LangChain`, `Anthropic`, or `Other`, then routes to a corresponding prompt chain.

Using a RunnableBranch

A `RunnableBranch` is initialized with a list of (condition, runnable) pairs and a default runnable. It selects which branch by passing each condition the input it's invoked with. It selects the first condition to evaluate to True, and runs the corresponding runnable to that condition with the input.

If no provided conditions match, it runs the default runnable.

Here's an example of what it looks like in action:

```
from langchain.prompts import PromptTemplate
from langchain_community.chat_models import
ChatAnthropic
from langchain_core.output_parsers import
StrOutputParser
```

First, let's create a chain that will identify incoming questions as being about `LangChain`, `Anthropic`, or `Other`:

```
chain = (
    PromptTemplate.from_template(
        """Given the user question below,
classify it as either being about
`LangChain`, `Anthropic`, or `Other`.
```

```
Do not respond with more than one word.

<question>
{question}
</question>

Classification:"""
    )
    | ChatAnthropic()
    | StrOutputParser()
)
```

```
chain.invoke({"question": "how do I call
Anthropic?"})
```

```
' Anthropic'
```

Now, let's create three sub chains:

```
langchain_chain = (
    PromptTemplate.from_template(
        """You are an expert in langchain. \
Always answer questions starting with "As
Harrison Chase told me". \
Respond to the following question:

Question: {question}
```

```
Answer:"""
    )
    | ChatAnthropic()
)
anthropic_chain = (
    PromptTemplate.from_template(
        """You are an expert in anthropic. \
Always answer questions starting with "As
Dario Amodei told me". \
Respond to the following question:

Question: {question}
Answer:"""
    )
    | ChatAnthropic()
)
general_chain = (
    PromptTemplate.from_template(
        """Respond to the following question:

Question: {question}
Answer:"""
    )
    | ChatAnthropic()
)
```

```
from langchain_core.runnables import
RunnableBranch

branch = RunnableBranch(
```

```
    (lambda x: "anthropic" in
x["topic"].lower(), anthropic_chain),
    (lambda x: "langchain" in
x["topic"].lower(), langchain_chain),
    general_chain,
)
```

```
full_chain = {"topic": chain, "question":
lambda x: x["question"]} | branch
```

```
full_chain.invoke({"question": "how do I use
Anthropic?"})
```

```
AIMessage(content=" As Dario Amodei told me,
here are some ways to use Anthropic:\n\n-
Sign up for an account on Anthropic's website
to access tools like Claude, Constitutional
AI, and Writer. \n\n- Use Claude for tasks
like email generation, customer service chat,
and QA. Claude can understand natural
language prompts and provide helpful
responses.\n\n- Use Constitutional AI if you
need an AI assistant that is harmless,
honest, and helpful. It is designed to be
safe and aligned with human values.\n\n- Use
Writer to generate natural language content
for things like marketing copy, stories,
```

```
reports, and more. Give it a topic and prompt
and it will create high-quality written
content.\n\n- Check out Anthropic's
documentation and blog for tips, tutorials,
examples, and announcements about new
capabilities as they continue to develop
their AI technology.\n\n- Follow Anthropic on
social media or subscribe to their newsletter
to stay up to date on new features and
releases.\n\n- For most people, the easiest
way to leverage Anthropic's technology is
through their website - just create an
account to get started!", additional_kwargs=
{}, example=False)
```

```
full_chain.invoke({"question": "how do I use
LangChain?"})
```

```
AIMessage(content=' As Harrison Chase told
me, here is how you use
LangChain:\n\nLangChain is an AI assistant
that can have conversations, answer
questions, and generate text. To use
LangChain, you simply type or speak your
input and LangChain will respond. \n\nYou can
ask LangChain questions, have discussions,
get summaries or explanations about topics,
and request it to generate text on a subject.
```

```
Some examples of interactions:\n\n- Ask general knowledge questions and LangChain will try to answer factually. For example "What is the capital of France?"\n\n- Have conversations on topics by taking turns speaking. You can prompt the start of a conversation by saying something like "Let\'s discuss machine learning"\n\n- Ask for summaries or high-level explanations on subjects. For example "Can you summarize the main themes in Shakespeare\'s Hamlet?" \n\n- Give creative writing prompts or requests to have LangChain generate text in different styles. For example "Write a short children\'s story about a mouse" or "Generate a poem in the style of Robert Frost about nature"\n\n- Correct LangChain if it makes an inaccurate statement and provide the right information. This helps train it.\n\nThe key is interacting naturally and giving it clear prompts and requests', additional_kwargs={}, example=False)
```

```
full_chain.invoke({"question": "whats 2 + 2"})
```

```
AIMessage(content=' 2 + 2 = 4', additional_kwargs={}, example=False)
```

Using a custom function

You can also use a custom function to route between different outputs. Here's an example:

```
def route(info):
    if "anthropic" in info["topic"].lower():
        return anthropic_chain
    elif "langchain" in
info["topic"].lower():
        return langchain_chain
    else:
        return general_chain
```

```
from langchain_core.runnables import
RunnableLambda

full_chain = {"topic": chain, "question":
lambda x: x["question"]} | RunnableLambda(
    route
)
```

```
full_chain.invoke({"question": "how do I use
Anthropic?"})
```

```
AIMessage(content=' As Dario Amodei told me,
to use Anthropic IPC you first need to import
it:\n\n```python\nfrom anthroipc import
ic\n```\n\nThen you can create a client and
connect to the server:\n\n```python \nclient
= ic.connect()\n```\n\nAfter that, you can
call methods on the client and get
responses:\n\n```python\nresponse =
client.ask("What is the meaning of
life?")\nprint(response)\n```\n\nYou can also
register callbacks to handle events:
\n\n```python\ndef on_poke(event):\n
print("Got poked!")\n\nclient.on(\'poke\',
on_poke)\n```\n\nAnd that\'s the basics of
using the Anthropic IPC client library for
Python! Let me know if you have any other
questions!', additional_kwargs={},
example=False)
```

```
full_chain.invoke({"question": "how do I use
LangChain?"})
```

```
AIMessage(content=' As Harrison Chase told me,
LangChain you first need to sign up for an API
platform.langchain.com. Once you have your API
can install the Python library and write a simp
script to call the LangChain API. Here is some
code to get started:\n\n```python\nimport
```

```
langchain\n\napi_key =
"YOUR_API_KEY"\n\nlangchain.set_key(api_key)\n\
= langchain.ask("What is the capital of
France?")\n\nprint(response.response)\n```\n\nT
send the question "What is the capital of Franc
the LangChain API and print the response. You c
customize the request by providing parameters l
max_tokens, temperature, etc. The LangChain Pyt
library documentation has more details on the a
options. The key things are getting an API key
calling langchain.ask() with your question text
know if you have any other questions!',
additional_kwargs={}, example=False)
```

```
full_chain.invoke({"question": "whats 2 +
2"})
```

```
AIMessage(content=' 4', additional_kwargs={},
example=False)
```