Configure chain internals at runtime

Oftentimes you may want to experiment with, or even expose to the end user, multiple different ways of doing things. In order to make this experience as easy as possible, we have defined two methods.

First, a `configurable_fields` method. This lets you configure particular fields of a runnable.

Second, a `configurable_alternatives` method. With this method, you can list out alternatives for any particular runnable that can be set during runtime.

Configuration Fields

With LLMs

With LLMs we can configure things like temperature

```
%pip install --upgrade --quiet  langchain
langchain-openai
```

```
from langchain.prompts import PromptTemplate
from langchain_core.runnables import
ConfigurableField
from langchain_openai import ChatOpenAI

model =
ChatOpenAI(temperature=0).configurable_fields(
    temperature=ConfigurableField(
        id="llm_temperature",
        name="LLM Temperature",
        description="The temperature of the
LLM",
    )
)
```

```
model.invoke("pick a random number")
```

```
AIMessage(content='7')
```

```
model.with_config(configurable=
{"llm_temperature": 0.9}).invoke("pick a
random number")
```

```
AIMessage(content='34')
```

We can also do this when its used as part of a chain

```
prompt = PromptTemplate.from_template("Pick a
random number above {x}")
chain = prompt | model
```

```
chain.invoke({"x": 0})
```

```
AIMessage(content='57')
```

```
chain.with_config(configurable=
{"llm_temperature": 0.9}).invoke({"x": 0})
```

```
AIMessage(content='6')
```

With HubRunnables

This is useful to allow for switching of prompts

```
from langchain.runnables.hub import
HubRunnable
```

```
prompt = HubRunnable("rlm/rag-
prompt").configurable_fields(
    owner_repo_commit=ConfigurableField(
        id="hub_commit",
        name="Hub Commit",
        description="The Hub commit to pull
from",
    )
)
```

```
prompt.invoke({"question": "foo", "context":
"bar"})
```

```
ChatPromptValue(messages=
[HumanMessage(content="You are an assistant
for question-answering tasks. Use the
following pieces of retrieved context to
answer the question. If you don't know the
answer, just say that you don't know. Use
three sentences maximum and keep the answer
concise.\nQuestion: foo \nContext: bar
\nAnswer:")])
```

```
prompt.with_config(configurable=
{"hub_commit": "rlm/rag-prompt-
llama"}).invoke(
    {"question": "foo", "context": "bar"}
)
```

```
ChatPromptValue(messages=
[HumanMessage(content="[INST]<<SYS>> You are
an assistant for question-answering tasks.
Use the following pieces of retrieved context
to answer the question. If you don't know the
answer, just say that you don't know. Use
three sentences maximum and keep the answer
concise.<</SYS>> \nQuestion: foo \nContext:
bar \nAnswer: [/INST]")])
```

Configurable Alternatives

With LLMs

Let’s take a look at doing this with LLMs

```
from langchain.prompts import PromptTemplate
from langchain_community.chat_models import
ChatAnthropic
from langchain_core.runnables import
```

```
ConfigurableField
from langchain_openai import ChatOpenAI
```

```
llm =
ChatAnthropic(temperature=0).configurable_alter
    # This gives this field an id
    # When configuring the end runnable, we can
use this id to configure this field
    ConfigurableField(id="llm"),
    # This sets a default_key.
    # If we specify this key, the default LLM
(ChatAnthropic initialized above) will be used
    default_key="anthropic",
    # This adds a new option, with name `openai
is equal to `ChatOpenAI()`
    openai=ChatOpenAI(),
    # This adds a new option, with name `gpt4`
equal to `ChatOpenAI(model="gpt-4")`
    gpt4=ChatOpenAI(model="gpt-4"),
    # You can add more configuration options he
)
prompt = PromptTemplate.from_template("Tell me 
about {topic}")
chain = prompt | llm
```

```
# By default it will call Anthropic
chain.invoke({"topic": "bears"})
```

```
AIMessage(content=" Here's a silly joke about bears:\n\nWhat do you call a bear with no teeth?\nA gummy bear!")
```

```
# We can use `.with_config(configurable={"llm": "openai"})` to specify an llm to use
chain.with_config(configurable={"llm": "openai"}).invoke({"topic": "bears"})
```

```
AIMessage(content="Sure, here's a bear joke for you:\n\nWhy don't bears wear shoes?\n\nBecause they already have bear feet!")
```

```
# If we use the `default_key` then it uses the default
chain.with_config(configurable={"llm": "anthropic"}).invoke({"topic": "bears"})
```

```
AIMessage(content=" Here's a silly joke about bears:\n\nWhat do you call a bear with no teeth?\nA gummy bear!")
```

With Prompts

We can do a similar thing, but alternate between prompts

```
llm = ChatAnthropic(temperature=0)
prompt = PromptTemplate.from_template(
    "Tell me a joke about {topic}"
).configurable_alternatives(
    # This gives this field an id
    # When configuring the end runnable, we
can then use this id to configure this field
    ConfigurableField(id="prompt"),
    # This sets a default_key.
    # If we specify this key, the default LLM
(ChatAnthropic initialized above) will be
used
    default_key="joke",
    # This adds a new option, with name
`poem`
    poem=PromptTemplate.from_template("Write
a short poem about {topic}"),
    # You can add more configuration options
here
)
chain = prompt | llm
```

```
# By default it will write a joke
chain.invoke({"topic": "bears"})
```

```
AIMessage(content=" Here's a silly joke about
bears:\n\nWhat do you call a bear with no
teeth?\nA gummy bear!")
```

```
# We can configure it write a poem
chain.with_config(configurable={"prompt":
"poem"}).invoke({"topic": "bears"})
```

```
AIMessage(content=' Here is a short poem
about bears:\n\nThe bears awaken from their
sleep\nAnd lumber out into the deep\nForests
filled with trees so tall\nForaging for food
before nightfall \nTheir furry coats and
claws so sharp\nSniffing for berries and fish
to nab\nLumbering about without a care\nThe
mighty grizzly and black bear\nProud
creatures, wild and free\nRuling their domain
majestically\nWandering the woods they call
their own\nBefore returning to their dens
alone')
```

With Prompts and LLMs

We can also have multiple things configurable! Here’s an example doing that with both prompts and LLMs.

```
llm =
ChatAnthropic(temperature=0).configurable_alter
    # This gives this field an id
    # When configuring the end runnable, we can
use this id to configure this field
    ConfigurableField(id="llm"),
    # This sets a default_key.
    # If we specify this key, the default LLM
(ChatAnthropic initialized above) will be used
    default_key="anthropic",
    # This adds a new option, with name `openai
is equal to `ChatOpenAI()`
    openai=ChatOpenAI(),
    # This adds a new option, with name `gpt4`
equal to `ChatOpenAI(model="gpt-4")`
    gpt4=ChatOpenAI(model="gpt-4"),
    # You can add more configuration options he
)
prompt = PromptTemplate.from_template(
    "Tell me a joke about {topic}"
).configurable_alternatives(
    # This gives this field an id
    # When configuring the end runnable, we can
use this id to configure this field
    ConfigurableField(id="prompt"),
    # This sets a default_key.
    # If we specify this key, the default LLM
(ChatAnthropic initialized above) will be used
    default_key="joke",
    # This adds a new option, with name `poem`
    poem=PromptTemplate.from_template("Write a
poem about {topic}"),
    # You can add more configuration options he
)
chain = prompt | llm
```

```
# We can configure it write a poem with
OpenAI
chain.with_config(configurable={"prompt":
"poem", "llm": "openai"}).invoke(
    {"topic": "bears"}
)
```

```
AIMessage(content="In the forest, where tall
trees sway,\nA creature roams, both fierce
and gray.\nWith mighty paws and piercing
eyes,\nThe bear, a symbol of strength,
defies.\n\nThrough snow-kissed mountains, it
does roam,\nA guardian of its woodland
home.\nWith fur so thick, a shield of
might,\nIt braves the coldest winter
night.\n\nA gentle giant, yet wild and
free,\nThe bear commands respect, you
see.\nWith every step, it leaves a trace,\nOf
untamed power and ancient grace.\n\nFrom
honeyed feast to salmon's leap,\nIt takes its
place, in nature's keep.\nA symbol of untamed
delight,\nThe bear, a wonder, day and
night.\n\nSo let us honor this noble
beast,\nIn forests where its soul finds
peace.\nFor in its presence, we come to
know,\nThe untamed spirit that in us also
flows.")
```

```
# We can always just configure only one if we
want
chain.with_config(configurable={"llm":
"openai"}).invoke({"topic": "bears"})
```

```
AIMessage(content="Sure, here's a bear joke
for you:\n\nWhy don't bears wear shoes?
\n\nBecause they have bear feet!")
```

Saving configurations

We can also easily save configured chains as their own objects

```
openai_poem = chain.with_config(configurable=
{"llm": "openai"})
```

```
openai_poem.invoke({"topic": "bears"})
```

```
AIMessage(content="Why don't bears wear
shoes?\n\nBecause they have bear feet!")
```