Prompt templates

Prompt templates are pre-defined recipes for generating prompts for language models.

A template may include instructions, few shot examples, and specific context and questions appropriate for a given task.

LangChain provides tooling to create and work with prompt templates.

LangChain strives to create model agnostic templates to make it easy to reuse existing templates across different language models.

Typically, language models expect the prompt to either be a string or else a list of chat messages.

Prompt template

Use `PromptTemplate` to create a template for a string prompt.

By default, `PromptTemplate` uses Python's str.format syntax for templating; however other templating syntax is available (e.g., `jinja2`).

```
from langchain import PromptTemplate

prompt_template = PromptTemplate.from_template(
    "Tell me a {adjective} joke about {content}."
)
prompt_template.format(adjective="funny", content="chickens")
```

```
"Tell me a funny joke about chickens."
```

The template supports any number of variables, including no variables:

```
from langchain import PromptTemplate

prompt_template = PromptTemplate.from_template(
"Tell me a joke"
```

```
)
prompt_template.format()
```

For additional validation, specify `input_variables` explicitly. These variables will be compared against the variables present in the template string during instantiation, raising an exception if there is a mismatch; for example,

```
from langchain import PromptTemplate

invalid_prompt = PromptTemplate(
    input_variables=["adjective"],
    template="Tell me a {adjective} joke about {content}."
)
```

You can create custom prompt templates that format the prompt in any way you want. For more information, see Custom Prompt Templates.

Chat prompt template

The prompt to Chat Models is a list of chat messages.

Each chat message is associated with content, and an additional parameter called `role`. For example, in the OpenAI Chat Completions API, a chat message can be associated with an AI assistant, a human or a system role.

Create a chat prompt template like this:

```
from langchain.prompts import ChatPromptTemplate

template = ChatPromptTemplate.from_messages([
    ("system", "You are a helpful AI bot. Your name is {name}."),
    ("human", "Hello, how are you doing?"),
    ("ai", "I'm doing well, thanks!"),
    ("human", "{user_input}"),
])

messages = template.format_messages(
    name="Bob",
    user_input="What is your name?"
)
```

`ChatPromptTemplate.from_messages` accepts a variety of message representations.

For example, in addition to using the 2-tuple representation of (type, content) used above, you could pass in an instance of `MessagePromptTemplate` or `BaseMessage`.

```
from langchain.prompts import ChatPromptTemplate
from langchain.prompts.chat import SystemMessage,
HumanMessagePromptTemplate

template = ChatPromptTemplate.from_messages(
    [
        SystemMessage(
            content=(
                "You are a helpful assistant that re-writes the user's
text to "
                "sound more upbeat."
            )
        ),
        HumanMessagePromptTemplate.from_template("{text}"),
    ]
)

from langchain.chat_models import ChatOpenAI

llm = ChatOpenAI()
llm(template.format_messages(text='i dont like eating tasty things.'))
```

```
AIMessage(content='I absolutely adore indulging in delicious treats!',
additional_kwargs={}, example=False)
```

This provides you with a lot of flexibility in how you construct your chat prompts.