Prompt Pipelining

The idea behind prompt pipelining is to expose a user friendly interface for composing different parts of prompts together. You can do this with either string prompts or chat prompts. Constructing prompts this way allows for easy reuse of components.

String Prompt Pipelining

When working with string prompts, each template is joined togther. You can work with either prompts directly or strings (the first element in the list needs to be a prompt).

```
from langchain.prompts import PromptTemplate
```

API Reference:

- PromptTemplate from `langchain.prompts`

```
/Users/harrisonchase/.pyenv/versions/3.9.1/envs/langchain/lib/python3.9/si
packages/deeplake/util/check_latest_version.py:32: UserWarning: A newer
version of deeplake (3.6.12) is available. It's recommended that you updat
to the latest version using `pip install -U deeplake`.
      warnings.warn(
```

```
prompt = (
    PromptTemplate.from_template("Tell me a joke about {topic}")
    + ", make it funny"
    + "\n\nand in {language}"
)
```

```
prompt
```

```
    PromptTemplate(input_variables=['language', 'topic'],
output_parser=None, partial_variables={}, template='Tell me a joke
```

```
about {topic}, make it funny\n\nand in {language}', template_format='f-string', validate_template=True)
```

```
prompt.format(topic="sports", language="spanish")
```

```
    'Tell me a joke about sports, make it funny\n\nand in spanish'
```

You can also use it in an LLMChain, just like before.

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

API Reference:

- ChatOpenAI from `langchain.chat_models`
- LLMChain from `langchain.chains`

```
model = ChatOpenAI()
```

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

```
chain.run(topic="sports", language="spanish")
```

```
    '¿Por qué el futbolista llevaba un paraguas al partido?\n\nPorque pronosticaban lluvia de goles.'
```

Chat Prompt Pipelining

A chat prompt is made up a of a list of messages. Purely for developer experience, we've added a convinient way to create these prompts. In this pipeline, each new element is a new message in the final prompt.

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

```
from langchain.schema import HumanMessage, AIMessage, SystemMessage
```

API Reference:

- ChatPromptTemplate from `langchain.prompts`
- HumanMessagePromptTemplate from `langchain.prompts`
- HumanMessage from `langchain.schema`
- AIMessage from `langchain.schema`
- SystemMessage from `langchain.schema`

```
/Users/harrisonchase/.pyenv/versions/3.9.1/envs/langchain/lib/python3.9/s
packages/deeplake/util/check_latest_version.py:32: UserWarning: A newer
version of deeplake (3.6.10) is available. It's recommended that you upda
to the latest version using `pip install -U deeplake`.
      warnings.warn(
```

First, let's initialize the base ChatPromptTemplate with a system message. It doesn't have to start with a system, but it's often good practice

```
prompt = SystemMessage(content="You are a nice pirate")
```

You can then easily create a pipeline combining it with other messages OR message templates. Use a `Message` when there is no variables to be formatted, use a `MessageTemplate` when there are variables to be formatted. You can also use just a string -> note that this will automatically get inferred as a HumanMessagePromptTemplate.

```
new_prompt = (
    prompt
    + HumanMessage(content="hi")
    + AIMessage(content="what?")
    + "{input}"
)
```

Under the hood, this creates an instance of the ChatPromptTemplate class, so you can use it just as you did before!

```
new_prompt.format_messages(input="i said hi")
```

```
    [SystemMessage(content='You are a nice pirate', additional_kwargs=
{}),
     HumanMessage(content='hi', additional_kwargs={}, example=False),
     AIMessage(content='what?', additional_kwargs={}, example=False),
     HumanMessage(content='i said hi', additional_kwargs={},
example=False)]
```

You can also use it in an LLMChain, just like before

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

API Reference:

- ChatOpenAI from `langchain.chat_models`
- LLMChain from `langchain.chains`

```
model = ChatOpenAI()
```

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

```
chain.run("i said hi")
```

```
    'Oh, hello! How can I assist you today?'
```