Prompts

Prompts for Chat models are built around messages, instead of just plain text.

You can make use of templating by using a `MessagePromptTemplate`. You can build a `ChatPromptTemplate` from one or more `MessagePromptTemplates`. You can use `ChatPromptTemplate`'s `format_prompt` -- this returns a `PromptValue`, which you can convert to a string or Message object, depending on whether you want to use the formatted value as input to an llm or chat model.

For convenience, there is a `from_template` method exposed on the template. If you were to use this template, this is what it would look like:

```
from langchain import PromptTemplate
from langchain.prompts.chat import (
    ChatPromptTemplate,
    SystemMessagePromptTemplate,
    AIMessagePromptTemplate,
    HumanMessagePromptTemplate,
)

template="You are a helpful assistant that translates {input_language}
to {output_language}."
system_message_prompt =
SystemMessagePromptTemplate.from_template(template)
human_template="{text}"
human_message_prompt =
HumanMessagePromptTemplate.from_template(human_template)
```

```
chat_prompt = ChatPromptTemplate.from_messages([system_message_prompt,
human_message_prompt])

# get a chat completion from the formatted messages
chat(chat_prompt.format_prompt(input_language="English",
output_language="French", text="I love programming.").to_messages())
```

```
    AIMessage(content="J'adore la programmation.", additional_kwargs=
{})
```

If you wanted to construct the MessagePromptTemplate more directly, you could create a PromptTemplate outside and then pass it in, eg:

```
prompt=PromptTemplate(
    template="You are a helpful assistant that translates {input_language} to {output_language}.",
    input_variables=["input_language", "output_language"],
)
system_message_prompt = SystemMessagePromptTemplate(prompt=prompt)
```