JinaChat

This notebook covers how to get started with JinaChat chat models.

```
from langchain.chat_models import JinaChat
from langchain.prompts.chat import (
    ChatPromptTemplate,
    SystemMessagePromptTemplate,
    AIMessagePromptTemplate,
    HumanMessagePromptTemplate,
)
from langchain.schema import AIMessage, HumanMessage, SystemMessage
```

```
chat = JinaChat(temperature=0)
```

```
messages = [
    SystemMessage(
        content="You are a helpful assistant that translates English to French."
    ),
    HumanMessage(
        content="Translate this sentence from English to French. I love programming."
    ),
]
chat(messages)
```

```
    AIMessage(content="J'aime programmer.", additional_kwargs={}, example=False)
```

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:

```
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'aime programmer.", additional_kwargs={}, example=False)
```