Few shot examples for chat models

This notebook covers how to use few shot examples in chat models. There does not appear to be solid consensus on how best to do few shot prompting, and the optimal prompt compilation will likely vary by model. Because of this, we provide few-shot prompt templates like the FewShotChatMessagePromptTemplate as a flexible starting point, and you can modify or replace them as you see fit.

The goal of few-shot prompt templates are to dynamically select examples based on an input, and then format the examples in a final prompt to provide for the model.

Note: The following code examples are for chat models. For similar few-shot prompt examples for completion models (LLMs), see the few-shot prompt templates guide.

Fixed Examples

The most basic (and common) few-shot prompting technique is to use a fixed prompt example. This way you can select a chain, evaluate it, and avoid worrying about additional moving parts in production.

The basic components of the template are:

- `examples`: A list of dictionary examples to include in the final prompt.
- `example_prompt`: converts each example into 1 or more messages through its `format_messages` method. A common example would be to convert each example into one human message and one AI message response, or a human message followed by a function call message.

Below is a simple demonstration. First, import the modules for this example:

```
from langchain.prompts import (
    FewShotChatMessagePromptTemplate,
    ChatPromptTemplate,
)
```

API Reference:

- FewShotChatMessagePromptTemplate from `langchain.prompts`
- ChatPromptTemplate from `langchain.prompts`

Then, define the examples you'd like to include.

```
examples = [
    {"input": "2+2", "output": "4"},
    {"input": "2+3", "output": "5"},
]
```

Next, assemble them into the few-shot prompt template.

```
# This is a prompt template used to format each individual example.
example_prompt = ChatPromptTemplate.from_messages(
    [
        ("human", "{input}"),
        ("ai", "{output}"),
    ]
)
few_shot_prompt = FewShotChatMessagePromptTemplate(
    example_prompt=example_prompt,
    examples=examples,
)

print(few_shot_prompt.format())
```

```
    Human: 2+2
    AI: 4
    Human: 2+3
    AI: 5
```

Finally, assemble your final prompt and use it with a model.

```
final_prompt = ChatPromptTemplate.from_messages(
    [
        ("system", "You are wonderous wizard of math."),
        few_shot_prompt,
        ("human", "{input}"),
    ]
)
```

```
from langchain.chat_models import ChatAnthropic

chain = final_prompt | ChatAnthropic(temperature=0.0)

chain.invoke({"input": "What's the square of a triangle?"})
```

API Reference:

- ChatAnthropic from `langchain.chat_models`

```
    AIMessage(content=' Triangles do not have a "square". A square refers
to a shape with 4 equal sides and 4 right angles. Triangles have 3 sides
and 3 angles.\n\nThe area of a triangle can be calculated using the
formula:\n\nA = 1/2 * b * h\n\nWhere:\n\nA is the area \nb is the base
(the length of one of the sides)\nh is the height (the length from the
base to the opposite vertex)\n\nSo the area depends on the specific
dimensions of the triangle. There is no single "square of a triangle". The
area can vary greatly depending on the base and height measurements.',
additional_kwargs={}, example=False)
```

Dynamic Few-shot Prompting

Sometimes you may want to condition which examples are shown based on the input. For this, you can replace the `examples` with an `example_selector`. The other components remain the same as above! To review, the dynamic few-shot prompt template would look like:

- `example_selector`: responsible for selecting few-shot examples (and the order in which they are returned) for a given input. These implement the BaseExampleSelector interface. A common example is the vectorstore-backed SemanticSimilarityExampleSelector
- `example_prompt`: convert each example into 1 or more messages through its `format_messages` method. A common example would be to convert each example into one

human message and one AI message response, or a human message followed by a function call message.

These once again can be composed with other messages and chat templates to assemble your final prompt.

```
from langchain.prompts import SemanticSimilarityExampleSelector
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import Chroma
```

API Reference:

- SemanticSimilarityExampleSelector from `langchain.prompts`
- OpenAIEmbeddings from `langchain.embeddings`
- Chroma from `langchain.vectorstores`

Since we are using a vectorstore to select examples based on semantic similarity, we will want to first populate the store.

```
examples = [
    {"input": "2+2", "output": "4"},
    {"input": "2+3", "output": "5"},
    {"input": "2+4", "output": "6"},
    {"input": "What did the cow say to the moon?", "output": "nothing at
all"},
    {
        "input": "Write me a poem about the moon",
        "output": "One for the moon, and one for me, who are we to talk
about the moon?",
    },
]

to_vectorize = [" ".join(example.values()) for example in examples]
embeddings = OpenAIEmbeddings()
vectorstore = Chroma.from_texts(to_vectorize, embeddings,
metadatas=examples)
```

Create the `example_selector`

With a vectorstore created, you can create the `example_selector`. Here we will isntruct it to only fetch the top 2 examples.

```
example_selector = SemanticSimilarityExampleSelector(
    vectorstore=vectorstore,
    k=2,
)

# The prompt template will load examples by passing the input do the
`select_examples` method
example_selector.select_examples({"input": "horse"})
```

```
    [{'input': 'What did the cow say to the moon?', 'output': 'nothing at
all'},
     {'input': '2+4', 'output': '6'}]
```

Create prompt template

Assemble the prompt template, using the `example_selector` created above.

```
from langchain.prompts import (
    FewShotChatMessagePromptTemplate,
    ChatPromptTemplate,
)

# Define the few-shot prompt.
few_shot_prompt = FewShotChatMessagePromptTemplate(
    # The input variables select the values to pass to the
example_selector
    input_variables=["input"],
    example_selector=example_selector,
    # Define how each example will be formatted.
    # In this case, each example will become 2 messages:
    # 1 human, and 1 AI
    example_prompt=ChatPromptTemplate.from_messages(
        [("human", "{input}"), ("ai", "{output}")]
    ),
)
```

API Reference:

- FewShotChatMessagePromptTemplate from `langchain.prompts`
- ChatPromptTemplate from `langchain.prompts`

Below is an example of how this would be assembled.

```
print(few_shot_prompt.format(input="What's 3+3?"))
```

```
    Human: 2+3
    AI: 5
    Human: 2+2
    AI: 4
```

Assemble the final prompt template:

```
final_prompt = ChatPromptTemplate.from_messages(
    [
        ("system", "You are wonderous wizard of math."),
        few_shot_prompt,
        ("human", "{input}"),
    ]
)
```

```
print(few_shot_prompt.format(input="What's 3+3?"))
```

```
    Human: 2+3
    AI: 5
    Human: 2+2
    AI: 4
```

Use with an LLM

Now, you can connect your model to the few-shot prompt.

```
from langchain.chat_models import ChatAnthropic

chain = final_prompt | ChatAnthropic(temperature=0.0)
```

```
chain.invoke({"input": "What's 3+3?"})
```

API Reference:

- ChatAnthropic from `langchain.chat_models`

```
    AIMessage(content=' 3 + 3 = 6', additional_kwargs={}, example=False)
```