



How to add Memory to an LLMChain

This notebook goes over how to use the Memory class with an LLMChain. For the purposes of this walkthrough, we will add the `ConversationBufferMemory` class, although this can be any memory class.

```
from langchain.chains import LLMChain
from langchain.llms import OpenAI
from langchain.memory import ConversationBufferMemory
from langchain.prompts import PromptTemplate
```



API Reference:

- `LLMChain` from `langchain.chains`
- `OpenAI` from `langchain.llms`
- `ConversationBufferMemory` from `langchain.memory`
- `PromptTemplate` from `langchain.prompts`

The most important step is setting up the prompt correctly. In the below prompt, we have two input keys: one for the actual input, another for the input from the Memory class. Importantly, we make sure the keys in the `PromptTemplate` and the `ConversationBufferMemory` match up (`chat_history`).

```
template = """You are a chatbot having a conversation with a human.

{chat_history}
Human: {human_input}
Chatbot: """

prompt = PromptTemplate(
    input_variables=["chat_history", "human_input"], template=template
)
memory = ConversationBufferMemory(memory_key="chat_history")
```

```
llm = OpenAI()
llm_chain = LLMChain(
    llm=llm,
    prompt=prompt,
```

```

        verbose=True,
        memory=memory,
    )

```

```
llm_chain.predict(human_input="Hi there my friend")
```

```

> Entering new LLMChain chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

```

```

Human: Hi there my friend
Chatbot:

```

```
> Finished chain.
```

```
' Hi there! How can I help you today?'
```

```
llm_chain.predict(human_input="Not too bad – how are you?")
```

```

> Entering new LLMChain chain...
Prompt after formatting:
You are a chatbot having a conversation with a human.

```

```

Human: Hi there my friend
AI: Hi there! How can I help you today?
Human: Not too bad – how are you?
Chatbot:

```

```
> Finished chain.
```

" I'm doing great, thanks for asking! How are you doing?"

Adding Memory to a Chat Model-based LLMChain

The above works for completion-style LLMs, but if you are using a chat model, you will likely get better performance using structured chat messages. Below is an example.

```
from langchain.chat_models import ChatOpenAI
from langchain.schema import SystemMessage
from langchain.prompts import ChatPromptTemplate,
HumanMessagePromptTemplate, MessagesPlaceholder
```

API Reference:

- `ChatOpenAI` from `langchain.chat_models`
- `SystemMessage` from `langchain.schema`
- `ChatPromptTemplate` from `langchain.prompts`
- `HumanMessagePromptTemplate` from `langchain.prompts`
- `MessagesPlaceholder` from `langchain.prompts`

We will use the `ChatPromptTemplate` class to set up the chat prompt.

The `from_messages` method creates a `ChatPromptTemplate` from a list of messages (e.g., `SystemMessage`, `HumanMessage`, `AIMessage`, `ChatMessage`, etc.) or message templates, such as the `MessagesPlaceholder` below.

The configuration below makes it so the memory will be injected to the middle of the chat prompt, in the "chat_history" key, and the user's inputs will be added in a human/user message to the end of the chat prompt.

```
prompt = ChatPromptTemplate.from_messages([
    SystemMessage(content="You are a chatbot having a conversation with a human."), # The persistent system prompt
    MessagesPlaceholder(variable_name="chat_history"), # Where the memory will be stored.
    HumanMessagePromptTemplate.from_template("{human_input}"), # Where the human input will be injected
])
```

```
memory = ConversationBufferMemory(memory_key="chat_history",
return_messages=True)
```

```
llm = ChatOpenAI()

chat_llm_chain = LLMChain(
    llm=llm,
    prompt=prompt,
    verbose=True,
    memory=memory,
)
```

```
chat_llm_chain.predict(human_input="Hi there my friend")
```

```
> Entering new LLMChain chain...
Prompt after formatting:
System: You are a chatbot having a conversation with a human.
Human: Hi there my friend
```

```
> Finished chain.
```

```
'Hello! How can I assist you today, my friend?'
```

```
chat_llm_chain.predict(human_input="Not too bad – how are you?")
```

```
> Entering new LLMChain chain...
Prompt after formatting:
System: You are a chatbot having a conversation with a human.
Human: Hi there my friend
AI: Hello! How can I assist you today, my friend?
Human: Not too bad – how are you?
```

> Finished chain.

"I'm an AI chatbot, so I don't have feelings, but I'm here to help and chat with you! Is there something specific you would like to talk about or any questions I can assist you with?"