How to add memory to a Multi-Input Chain

Most memory objects assume a single input. In this notebook, we go over how to add memory to a chain that has multiple inputs. As an example of such a chain, we will add memory to a question/answering chain. This chain takes as inputs both related documents and a user question.

```
from langchain.embeddings.openai import OpenAIEmbeddings
from langchain.embeddings.cohere import CohereEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores.elastic_vector_search import
ElasticVectorSearch
from langchain.vectorstores import Chroma
from langchain.docstore.document import Document
```

API Reference:

- OpenAIEmbeddings from `langchain.embeddings.openai`
- CohereEmbeddings from `langchain.embeddings.cohere`
- CharacterTextSplitter from `langchain.text_splitter`
- ElasticVectorSearch from `langchain.vectorstores.elastic_vector_search`
- Chroma from `langchain.vectorstores`
- Document from `langchain.docstore.document`

```
with open("../../state_of_the_union.txt") as f:
    state_of_the_union = f.read()
text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=0)
texts = text_splitter.split_text(state_of_the_union)

embeddings = OpenAIEmbeddings()
```

```
docsearch = Chroma.from_texts(
    texts, embeddings, metadatas=[{"source": i} for i in
range(len(texts))]
)
```

```
    Running Chroma using direct local API.
    Using DuckDB in-memory for database. Data will be transient.
```

```
query = "What did the president say about Justice Breyer"
docs = docsearch.similarity_search(query)
```

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

API Reference:

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

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

Given the following extracted parts of a long document and a question,
create a final answer.

{context}

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

prompt = PromptTemplate(
    input_variables=["chat_history", "human_input", "context"],
template=template
)
memory = ConversationBufferMemory(memory_key="chat_history",
input_key="human_input")
chain = load_qa_chain(
    OpenAI(temperature=0), chain_type="stuff", memory=memory,
prompt=prompt
)
```

```
query = "What did the president say about Justice Breyer"
chain({"input_documents": docs, "human_input": query},
return_only_outputs=True)
```

```
    {'output_text': ' Tonight, I'd like to honor someone who has
dedicated his life to serve this country: Justice Stephen Breyer—an
Army veteran, Constitutional scholar, and retiring Justice of the
United States Supreme Court. Justice Breyer, thank you for your
service.'}
```

```
print(chain.memory.buffer)
```

```
    Human: What did the president say about Justice Breyer
    AI:  Tonight, I'd like to honor someone who has dedicated his life
to serve this country: Justice Stephen Breyer—an Army veteran,
Constitutional scholar, and retiring Justice of the United States
Supreme Court. Justice Breyer, thank you for your service.
```