LangChain: Memory

Outline

- ConversationBufferMemory
- ConversationBufferWindowMemory
- ConversationTokenBufferMemory
- ConversationSummaryMemory

ConversationBufferMemory

In [72]:
```
import os

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file

import warnings
warnings.filterwarnings('ignore')
```

In [73]:
```
from langchain.chat_models import ChatOpenAI
from langchain.chains import ConversationChain
from langchain.memory import ConversationBufferMemory
```

In [74]:
```
llm = ChatOpenAI(temperature=0.0)
memory = ConversationBufferMemory()
conversation = ConversationChain(
    llm=llm,
    memory = memory,
    verbose=True
)
```

```
In [75]: conversation.predict(input="Hi, my name is Andrew")
```

```
> Entering new ConversationChain chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is
talkative and provides lots of specific details from its context. If the AI d
oes not know the answer to a question, it truthfully says it does not know.

Current conversation:

Human: Hi, my name is Andrew
AI:

> Finished chain.

"Hello Andrew, it's nice to meet you. My name is AI. How can I assist you tod
ay?"
```

```
In [76]: conversation.predict(input="What is 1+1?")
```

```
> Entering new ConversationChain chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is
talkative and provides lots of specific details from its context. If the AI d
oes not know the answer to a question, it truthfully says it does not know.

Current conversation:
Human: Hi, my name is Andrew
AI: Hello Andrew, it's nice to meet you. My name is AI. How can I assist you
today?
Human: What is 1+1?
AI:

> Finished chain.

'The answer to 1+1 is 2.'
```

In [77]:
```
conversation.predict(input="What is my name?")
```

```
> Entering new ConversationChain chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is
talkative and provides lots of specific details from its context. If the AI d
oes not know the answer to a question, it truthfully says it does not know.

Current conversation:
Human: Hi, my name is Andrew
AI: Hello Andrew, it's nice to meet you. My name is AI. How can I assist you
today?
Human: What is 1+1?
AI: The answer to 1+1 is 2.
Human: What is my name?
AI:

> Finished chain.

'Your name is Andrew, as you mentioned earlier.'
```

In [78]:
```
print(memory.buffer)
```

```
Human: Hi, my name is Andrew
AI: Hello Andrew, it's nice to meet you. My name is AI. How can I assist you
today?
Human: What is 1+1?
AI: The answer to 1+1 is 2.
Human: What is my name?
AI: Your name is Andrew, as you mentioned earlier.
```

In [79]:
```
memory.load_memory_variables({})
```

```
{'history': "Human: Hi, my name is Andrew\nAI: Hello Andrew, it's nice to mee
t you. My name is AI. How can I assist you today?\nHuman: What is 1+1?\nAI: T
he answer to 1+1 is 2.\nHuman: What is my name?\nAI: Your name is Andrew, as
you mentioned earlier."}
```

In [80]:
```
memory = ConversationBufferMemory()
```

In [81]:
```
memory.save_context({"input": "Hi"},
                    {"output": "What's up"})
```

In [82]:
```
print(memory.buffer)
```

```
Human: Hi
AI: What's up
```

In [83]:
```
memory.load_memory_variables({})
```

```
{'history': "Human: Hi\nAI: What's up"}
```

In [84]:
```
memory.save_context({"input": "Not much, just hanging"},
                    {"output": "Cool"})
```

In [85]:
```
memory.load_memory_variables({})
```

```
{'history': "Human: Hi\nAI: What's up\nHuman: Not much, just hanging\nAI: Coo
l"}
```

ConversationBufferWindowMemory

In [86]:
```
from langchain.memory import ConversationBufferWindowMemory
```

In [87]:
```
memory = ConversationBufferWindowMemory(k=1)
```

In [88]:
```
memory.save_context({"input": "Hi"},
                    {"output": "What's up"})
memory.save_context({"input": "Not much, just hanging"},
                    {"output": "Cool"})
```

In [89]:
```
memory.load_memory_variables({})
```

```
{'history': 'Human: Not much, just hanging\nAI: Cool'}
```

In [90]:
```
llm = ChatOpenAI(temperature=0.0)
memory = ConversationBufferWindowMemory(k=1)
conversation = ConversationChain(
    llm=llm,
    memory = memory,
    verbose=False
)
```

In [91]:
```
conversation.predict(input="Hi, my name is Andrew")
```

```
"Hello Andrew, it's nice to meet you. My name is AI. How can I assist you tod
ay?"
```

In [92]:
```
conversation.predict(input="What is 1+1?")
```

```
'The answer to 1+1 is 2.'
```

In [93]:
```
conversation.predict(input="What is my name?")
```

```
"I'm sorry, I don't have access to that information. Could you please tell me
your name?"
```

ConversationTokenBufferMemory

In [94]:
```
#!pip install tiktoken
```

In [95]:
```
from langchain.memory import ConversationTokenBufferMemory
from langchain.llms import OpenAI
llm = ChatOpenAI(temperature=0.0)
```

In [96]:
```
memory = ConversationTokenBufferMemory(llm=llm, max_token_limit=30)
memory.save_context({"input": "AI is what?!"},
                    {"output": "Amazing!"})
memory.save_context({"input": "Backpropagation is what?"},
                    {"output": "Beautiful!"})
memory.save_context({"input": "Chatbots are what?"},
                    {"output": "Charming!"})
```

In [97]:
```
memory.load_memory_variables({})
```

```
{'history': 'AI: Beautiful!\nHuman: Chatbots are what?\nAI: Charming!'}
```

ConversationSummaryMemory

In [98]:
```
from langchain.memory import ConversationSummaryBufferMemory
```

In [99]:
```
# create a long string
schedule = "There is a meeting at 8am with your product team. \
You will need your powerpoint presentation prepared. \
9am-12pm have time to work on your LangChain \
project which will go quickly because Langchain is such a powerful tool. \
At Noon, lunch at the italian resturant with a customer who is driving \
from over an hour away to meet you to understand the latest in AI. \
Be sure to bring your laptop to show the latest LLM demo."

memory = ConversationSummaryBufferMemory(llm=llm, max_token_limit=100)
memory.save_context({"input": "Hello"}, {"output": "What's up"})
memory.save_context({"input": "Not much, just hanging"},
                    {"output": "Cool"})
memory.save_context({"input": "What is on the schedule today?"},
                    {"output": f"{schedule}"})
```

In [100]:
```
memory.load_memory_variables({})
```

```
{'history': "System: The human and AI engage in small talk before discussing
the day's schedule. The AI informs the human of a morning meeting with the pr
oduct team, time to work on the LangChain project, and a lunch meeting with a
customer interested in the latest AI developments."}
```

In [101]:
```
conversation = ConversationChain(
    llm=llm,
    memory = memory,
    verbose=True
)
```

In [102]:
```
conversation.predict(input="What would be a good demo to show?")
```

```
> Entering new ConversationChain chain...
Prompt after formatting:
The following is a friendly conversation between a human and an AI. The AI is
talkative and provides lots of specific details from its context. If the AI d
oes not know the answer to a question, it truthfully says it does not know.

Current conversation:
System: The human and AI engage in small talk before discussing the day's sch
edule. The AI informs the human of a morning meeting with the product team, t
ime to work on the LangChain project, and a lunch meeting with a customer int
erested in the latest AI developments.
Human: What would be a good demo to show?
AI:

> Finished chain.

"Based on the customer's interest in AI developments, I would suggest showcas
ing our latest natural language processing capabilities. We could demonstrate
how our AI can understand and respond to complex language queries, and even p
rovide personalized recommendations based on the user's preferences. Addition
ally, we could highlight our machine learning algorithms and how they continu
ously improve the accuracy and efficiency of our AI. Would you like me to pre
pare a demo for the meeting?"
```

In [103]:
```
memory.load_memory_variables({})
```

```
{'history': "System: The human and AI engage in small talk before discussing
the day's schedule. The AI informs the human of a morning meeting with the pr
oduct team, time to work on the LangChain project, and a lunch meeting with a
customer interested in the latest AI developments. The human asks what would
be a good demo to show.\nAI: Based on the customer's interest in AI developme
nts, I would suggest showcasing our latest natural language processing capabi
lities. We could demonstrate how our AI can understand and respond to complex
language queries, and even provide personalized recommendations based on the
user's preferences. Additionally, we could highlight our machine learning alg
orithms and how they continuously improve the accuracy and efficiency of our
AI. Would you like me to prepare a demo for the meeting?"}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: