# 会话记忆和窗口记忆

在 LangChain 中，我们需要导入所需的记忆类型，然后实例化，最后传递给链（Chain），完成了模型的记忆功能。

为了比较ConversationBufferMemory 和 ConversationBufferWindowMemory 之间的区别。我们先导入公共的代码。

#### 公共代码

先安装库：

!pip -q install openai langchain

设置密钥：

import os  
os.environ['OPENAI\_API\_KEY'] = ''

引入各组件，实例化一个会话链（ConversationChain）。 这里我们使用的 Chain 只是一个简单的对话链 ConversationChain，允许我们跟 OpenAI 模型交互，并传递我们想要说的内容。：

from langchain import OpenAI  
llm = OpenAI(model\_name='text-davinci-003',   
 temperature=0,   
 max\_tokens = 256)  
from langchain.chains import ConversationChain

#### 缓冲区记忆代码

我们先看对话缓冲区记忆, 这里先导入且实例化ConversationBufferMemory组件。

from langchain.chains.conversation.memory import ConversationBufferMemory  
memory = ConversationBufferMemory()

让我们开始对话，每次输入后等待AI返回的信息后，再输下一条：

# 请依次运行以下代码，不要一次性运行。  
conversation.predict(input="你好，我叫美丽")  
conversation.predict(input="今天心情怎么样？")  
conversation.predict(input="我想找客户服务中心")  
conversation.predict(input="我的洗衣机坏了")

执行完最后一条对话后，我们看到的会话链显示：

'> Entering new 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 does not know the answer to a question, it truthfully says it does not know.  
  
Current conversation:  
Human: 你好，我叫美丽  
AI: 你好，美丽！很高兴认识你！我是一个AI，我可以回答你的问题，也可以与你聊天。你想问我什么？  
Human: 今天心情怎么样？  
AI: 今天我的心情很好！我很开心能够和你聊天！  
Human: 我想找客户服务中心  
AI: 好的，我可以帮助你找到客户服务中心。你知道客户服务中心在哪里吗？  
Human: 我的洗衣机坏了  
AI: 哦，很抱歉听到你的洗衣机坏了。你可以联系客户服务中心来获得帮助。你知道客户服务中心的联系方式吗？  
Human: 我的洗衣机坏了  
AI:  
  
'> Finished chain.

你会看到，Current conversation 内部把所有人类和AI的对话记录都保存了。这样做可以让你看到我们之前对话的确切内容，这是 LangChain 中最简单的记忆形式，但仍然非常强大，特别是当你知道人们与你的聊天的互动次数有限，或者你实际上要在五次互动后关闭它，诸如此类的情况下，这种记忆将非常有效。

#### 窗口缓冲区记忆

运行完公共代码后，再导入且实例化窗口缓冲区记忆（ConversationBufferWindowMemory）组件。

from langchain.chains.conversation.memory import ConversationBufferWindowMemory  
memory = ConversationBufferWindowMemory(k=2)  
  
conversation = ConversationChain(  
 llm=llm,   
 verbose=True,   
 memory=memory  
)

让我们开始对话，每次输入后等待AI返回的信息后，再输下一条：

# 请依次运行以下代码，不要一次性运行。  
conversation.predict(input="你好，我叫美丽")  
conversation.predict(input="今天心情怎么样？")  
conversation.predict(input="我想找客户服务中心")  
conversation.predict(input="我的洗衣机坏了")

执行完最后一条对话后，我们看到的会话链显示：

'> Entering new 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 does not know the answer to a question, it truthfully says it does not know.  
  
Current conversation:  
Human: 今天心情怎么样？  
AI: 今天我的心情很好！我很开心能够和你聊天！  
Human: 我想找客户服务中心  
AI: 好的，我可以帮助你找到客户服务中心。你知道客户服务中心在哪里吗？  
Human: 我的洗衣机坏了  
AI:  
  
'> Finished chain.

#### 跟缓冲区记忆的区别在哪里？

为了对比两种类型的区别，我们采用的对话的内容是一样，但是由于我们设置了k = 2，但你实际上可以将其设置得比这个高得多，可以获取最后 5 次或 10 次互动，具体取决于你想使用多少个标记。

最后打印的Current conversation部分，我们可以看到最初打招呼，介绍我自己的信息和AI回应并没有记忆下来。它 **丢失** 了“你好，我叫美丽”这句话，它只将最后两次互动传递给大型语言模型。

所以如果你有一些情况，可以将其设置为 k = 5 或者 k = 10 ，大多数对话可能不会有很大变化。实际上缓冲区窗口记忆比缓冲区记忆多了限制，不会记录所有人和AI的会话记录,而是根据配置K的数值来对话记录条目，实现控制提示长度的目的。