多轮对话中让AI保持长期记忆的8种优化方式篇

来自： AiGC面试宝典

宁静致远

2024年02月08日 10:05

一、前言

在基于大模型的 Agent 中，长期记忆的状态维护至关重要，在 OpenAI AI 应用研究主管 Lilian Weng 的博客《基于大模型的 Agent 构成》[1]中，将记忆视为关键的组件之一，下面我将结合 LangChain 中的代码，8 种不同的记忆维护方式在不同场景中的应用。

二、Agent 如何获取上下文对话信息?

2.1 获取全量历史对话

以一般客服场景为例
在电信公司的客服聊天机器人场景中，如果用户在对话中先是询问了账单问题，接着又谈到了网络连接问题，ConversationBufferMemory 可以用来记住整个与用户的对话历史，可以帮助 AI 在回答网络问题时还记得账单问题的相关细节，从而提供更连贯的服务。

```
from langchain.memory import ConversationBufferMemory
memory = ConversationBufferMemory()
memory.save_context({"input": "你好"}, {"output": "怎么了"})

variables = memory.load_memory_variables({})
```

2.2 滑动窗口获取最近部分对话内容

以商品咨询场景为例
在一个电商平台上，如果用户询问关于特定产品的问题（如手机的电池续航时间），然后又问到了配送方式，ConversationBufferWindowMemory 可以帮助 AI 只专注于最近的一两个问题（如配送方式），而不是整个对话历史，以提供更快速和专注的答复。

```
from langchain.memory import ConversationBufferWindowMemory

# 只保留最后1次互动的记忆
memory = ConversationBufferWindowMemory(k=1)
```

2.3 获取历史对话中实体信息

以法律咨询场景为例

在法律咨询的场景中，客户可能会提到特定的案件名称、相关法律条款或个人信息（如“我在去年的交通事故中受了伤，想了解关于赔偿的法律建议”）。ConversationEntityMemory 可以帮助 AI 记住这些关键实体和实体关系细节，从而在整个对话过程中提供更准确、更个性化的法律建议。

```
llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)
memory = ConversationEntityMemory(llm=llm)
_input = {"input": "公众号《LLM应用全栈开发》的作者是莫尔索"}
memory.load_memory_variables(_input)
memory.save_context(
    _input,
    {"output": "是吗，这个公众号是干嘛的"}
)
print(memory.load_memory_variables({"input": "莫尔索是谁？"}))
# 输出，可以看到提取了实体关系
{'history': 'Human: 公众号《LLM应用全栈开发》的作者是莫尔索 \n AI: 是吗，这个公众号是干嘛的',
'entities': {'莫尔索': '《LLM应用全栈开发》的作者。'}}
```

2.4 利用知识图谱获取历史对话中的实体及其联系

以医疗咨询场景为例

在医疗咨询中，一个病人可能会描述多个症状和过去的医疗历史（如“我有糖尿病史，最近觉得经常口渴和疲劳”）。ConversationKGMemory 可以构建一个包含病人症状、疾病历史和可能的健康关联的知识图谱，从而帮助 AI 提供更全面和深入的医疗建议。

```
from langchain.memory import ConversationKGMemory
from langchain.llms import OpenAI
llm = OpenAI(temperature=0)
memory = ConversationKGMemory(llm=llm)
memory.save_context({"input": "小李是程序员"}, {"output": "知道了，小李是程序员"})
memory.save_context({"input": "莫尔索是小李的笔名"}, {"output": "明白，莫尔索是小李的笔名"})

variables = memory.load_memory_variables({"input": "告诉我关于小李的信息"})
print(variables)
# 输出
{'history': 'On 小李: 小李 is 程序员. 小李 的笔名 莫尔索.'}
```

2.5 对历史对话进行阶段性总结摘要

以教育辅导场景为例

在一系列的教育辅导对话中，学生可能会提出不同的数学问题或理解难题（如“我不太理解二次方程的求解方法”）。ConversationSummaryMemory 可以帮助 AI 总结之前的辅导内容和学生的疑问点，以便在随后的辅导中提供更针对性的解释和练习.

2.6 需要获取最新对话，又要兼顾较早历史对话

以技术支持场景为例

在处理一个长期的技术问题时（如软件故障排查），用户可能会在多次对话中提供不同的错误信息和反馈。ConversationSummaryBufferMemory 可以帮助 AI 保留最近几次交互的详细信息，同时提供历史问题处理的摘要，以便于更有效地识别和解决问题。

2.7 回溯最近和最关键的对话信息

以金融咨询场景为例

在金融咨询聊天机器人中，客户可能会提出多个问题，涉及投资、市场动态或个人财务规划（如“我想了解股市最近的趋势以及如何分配我的投资组合”）。ConversationTokenBufferMemory 可以帮助 AI 聚焦于最近和最关键的几个问题，同时避免由于记忆过多而导致的信息混淆。

2.8 基于向量检索对话信息

以了解最新新闻事件为例
用户可能会对特定新闻事件提出问题，如“最近的经济峰会有什么重要决策？”VectorStoreRetrieverMemory 能够快速从大量历史新闻数据中检索出与当前问题最相关的信息，即使这些信息在整个对话历史中不是最新的，也能提供及时准确的背景信息和详细报道。

```
vectorstore = Chroma(embedding_function=OpenAIEmbeddings())
retriever = vectorstore.as_retriever(search_kwargs=dict(k=1))
memory = VectorStoreRetrieverMemory(retriever=retriever)

memory.save_context({"input": "我喜欢吃火锅"}, {"output": "听起来很好吃"})
memory.save_context({"input": "我不喜欢看摔跤比赛"}, {"output": "我也是"})

PROMPT_TEMPLATE = """以下是人类和 AI 之间的友好对话。AI 话语多且提供了许多来自其上下文的具体细节。如果 AI 不知道问题的答案，它会诚实地说不知道。

以前对话的相关片段：
{history}

（如果不相关，你不需要使用这些信息）

当前对话：
人类：{input}
AI：
"""

prompt = PromptTemplate(input_variables=["history", "input"], template=PROMPT_TEMPLATE)
conversation_with_summary = ConversationChain(
    llm=llm,
    prompt=prompt,
    memory=memory,
    verbose=True
)

print(conversation_with_summary.predict(input="你好，我是莫尔索，你叫什么"))
print(conversation_with_summary.predict(input="我喜欢的食物是什么？"))
print(conversation_with_summary.predict(input="我提到了哪些运动？"))
```