# 自定义记忆组件

虽然LangChain中预设了几种类型的记忆组件，但在实际应用中，你可能需要根据自己的应用场景来添加自定义的记忆类型。

在本节中，我们将向ConversationChain添加一个自定义的记忆类型。为了添加自定义的记忆类，我们需要引入基础记忆类并对其进行子类化。

from langchain import OpenAI, ConversationChain  
from langchain.schema import BaseMemory  
from pydantic import BaseModel  
from typing import List, Dict, Any

在这个例子中，我们将编写一个自定义记忆类，它使用spacy来提取实体，并将有关它们的信息保存在一个简单的哈希表中。然后，在对话中，我们将观察输入文本，提取任何实体，并将关于它们的任何信息放入上下文中。

请注意，这种实现方式相当简单且脆弱，可能在生产环境中并不实用。它的目的是展示你可以添加自定义记忆实现。为此，我们需要spacy。

!pip install spacy  
!python -m spacy download en\_core\_web\_lg

import spacy  
  
nlp = spacy.load("en\_core\_web\_lg")

class SpacyEntityMemory(BaseMemory, BaseModel):  
 """Memory class for storing information about entities."""  
  
 # Define dictionary to store information about entities.  
 entities: dict = {}  
 # Define key to pass information about entities into prompt.  
 memory\_key: str = "entities"  
  
 def clear(self):  
 self.entities = {}  
  
 @property  
 def memory\_variables(self) -> List[str]:  
 """Define the variables we are providing to the prompt."""  
 return [self.memory\_key]  
  
 def load\_memory\_variables(self, inputs: Dict[str, Any]) -> Dict[str, str]:  
 """Load the memory variables, in this case the entity key."""  
 # Get the input text and run through spacy  
 doc = nlp(inputs[list(inputs.keys())[0]])  
 # Extract known information about entities, if they exist.  
 entities = [  
 self.entities[str(ent)] for ent in doc.ents if str(ent) in self.entities  
 ]  
 # Return combined information about entities to put into context.  
 return {self.memory\_key: "\n".join(entities)}  
  
 def save\_context(self, inputs: Dict[str, Any], outputs: Dict[str, str]) -> None:  
 """Save context from this conversation to buffer."""  
 # Get the input text and run through spacy  
 text = inputs[list(inputs.keys())[0]]  
 doc = nlp(text)  
 # For each entity that was mentioned, save this information to the dictionary.  
 for ent in doc.ents:  
 ent\_str = str(ent)  
 if ent\_str in self.entities:  
 self.entities[ent\_str] += f"\n{text}"  
 else:  
 self.entities[ent\_str] = text

我们现在定义一个提示，它接受关于实体的信息以及用户输入的信息。

from langchain.prompts.prompt import PromptTemplate  
  
template = """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. You are provided with information about entities the Human mentions, if relevant.  
  
Relevant entity information:  
{entities}  
  
Conversation:  
Human: {input}  
AI:"""  
prompt = PromptTemplate(input\_variables=["entities", "input"], template=template)

然后我们组合在一起。

llm = OpenAI(temperature=0)  
conversation = ConversationChain(  
 llm=llm, prompt=prompt, verbose=True, memory=SpacyEntityMemory()  
)

在第一个例子中，由于没有关于Harrison的先验知识，“Relevant entity information”（相关实体信息）部分是空的。

conversation.predict(input="Harrison likes machine learning")

> 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 does not know the answer to a question, it truthfully says it does not know. You are provided with information about entities the Human mentions, if relevant.  
   
 Relevant entity information:  
   
   
 Conversation:  
 Human: Harrison likes machine learning  
 AI:  
   
 > Finished ConversationChain chain.  
  
  
  
  
  
 " That's great to hear! Machine learning is a fascinating field of study. It involves using algorithms to analyze data and make predictions. Have you ever studied machine learning, Harrison?"

现在在第二个例子中，我们可以看到它抽取了关于Harrison的信息。

conversation.predict(  
 input="What do you think Harrison's favorite subject in college was?"  
)

> 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 does not know the answer to a question, it truthfully says it does not know. You are provided with information about entities the Human mentions, if relevant.  
   
 Relevant entity information:  
 Harrison likes machine learning  
   
 Conversation:  
 Human: What do you think Harrison's favorite subject in college was?  
 AI:  
   
 > Finished ConversationChain chain.  
  
  
  
  
  
 ' From what I know about Harrison, I believe his favorite subject in college was machine learning. He has expressed a strong interest in the subject and has mentioned it often.'

请再次注意，这种实现方式相当简单且脆弱，可能在生产环境中并不实用。它的目的是为了展示你可以添加自定义的记忆实现。