# 2.3.2 Pydantic JSON解析器

JSON {} 对象，这种格式最大的特点是人和机器都看得懂。

你可以把JSON对象想象成一个大家都认识的“信息盒子”。在这个“信息盒子”里，我们可以存储各种各样的信息，比如你的名字、你的年龄、你最喜欢的食物，甚至是你所有玩具的列表等等。这些信息都被整齐地放在“信息盒子”里，每一种信息都有自己的标签，比如“名字”、“年龄”、“食物”、“玩具”。

在我们开发语言模型应用的时候，我们经常用到这个“信息盒子”。因为它可以帮我们更好地整理和使用机器人的答案。比如，机器人可能会给我们一个包含很多信息的答案，而我们可以用这个“信息盒子”来把这些信息整理得更清晰，更易于理解和使用。

所以，JSON对象就像一个非常有用的“信息盒子”，可以帮助我们更好地使用和理解语言模型的答案。

请记住，大语言模型是有“缺陷”的抽象！你需要使用一个具有足够能力的模型来生成格式良好的JSON。在OpenAI 模型家族中，DaVinci可以做到这一点，但Curie的能力已经大幅下降。Langchain 这种输出解析器可以指定一个任意的JSON结构，并向大语言模型查询，输出符合该架构的JSON。

你可以使用Pydantic来声明你的数据模型。Pydantic的BaseModel就像一个Python数据类，但它具有实际的类型检查和强制转换功能。

下面是最简单的Pydantic JSON解析器代码：

导入语言模型 OpenAI和Prompts模板。

from langchain.prompts import (  
 PromptTemplate,  
 ChatPromptTemplate,  
 HumanMessagePromptTemplate,  
)  
from langchain.llms import OpenAI  
from langchain.chat\_models import ChatOpenAI

获取格式化指令 PydanticOutputParser:

from langchain.output\_parsers import PydanticOutputParser  
from pydantic import BaseModel, Field, validator  
from typing import List

这里我们引入 OpenAI 的模型 text-davinci-003。

model\_name = "text-davinci-003"  
temperature = 0.0  
model = OpenAI(model\_name=model\_name, temperature=temperature)

定义你想要的数据格式：

# Define your desired data structure.  
class Joke(BaseModel):  
 setup: str = Field(description="question to set up a joke")  
 punchline: str = Field(description="answer to resolve the joke")  
  
 # You can add custom validation logic easily with Pydantic.  
 @validator("setup")  
 def question\_ends\_with\_question\_mark(cls, field):  
 if field[-1] != "?":  
 raise ValueError("Badly formed question!")  
 return field  
  
  
# And a query intented to prompt a language model to populate the data structure.  
joke\_query = "Tell me a joke."  
  
# Set up a parser + inject instructions into the prompt template.  
parser = PydanticOutputParser(pydantic\_object=Joke)  
  
prompt = PromptTemplate(  
 template="Answer the user query.\n{format\_instructions}\n{query}\n",  
 input\_variables=["query"],  
 partial\_variables={"format\_instructions": parser.get\_format\_instructions()},  
)  
  
\_input = prompt.format\_prompt(query=joke\_query)  
  
output = model(\_input.to\_string())  
  
parser.parse(output)

将提示传入OpenAI 模型:

\_input = prompt.format(subject="ice cream flavors")  
output = model(\_input)

调用解析器的parse方法,解析数据为列表格式.

output\_parser.parse(output)

最终的结果是:

Joke(setup='Why did the chicken cross the road?', punchline='To get to the other side!')

细心的读者会注意到,我们这次仍旧是使用了输出解析器的2大方法: 格式化 output\_parser.get\_format\_instructions() 和 解析 output\_parser.parse()。

掌握这两种方法，我们就能掌握输出解析器的使用方法。