OpenAI Function Calling

Notes:

- LLM's don't always produce the same results. The results you see in this notebook may differ from the results you see in the video.
- Notebooks results are temporary. Download the notebooks to your local machine if you wish to save your results.

In [80]:
```
import os
import openai

from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file
openai.api_key = os.environ['OPENAI_API_KEY']
```

Making function that will return the wheater value

In [81]:
```
import json

# Example dummy function hard coded to return the same weather
# In production, this could be your backend API or an external API
def get_current_weather(location, unit="fahrenheit"):
    """Get the current weather in a given location"""
    weather_info = {
        "location": location,
        "temperature": "72",
        "unit": unit,
        "forecast": ["sunny", "windy"],
    }
    return json.dumps(weather_info)
```

So now we will use new feature of openai which is "functions" to specify that it has to hit above function.

In [82]:
```
# define a function
functions = [
    {
        #name of the function that will be called
        "name": "get_current_weather",

        #this description is important because it will be passed to LLM
        "description": "Get the current weather in a given location",


        #************************************************************
        #these parameters will be obtained from the prompot and properties and desciption of properties will help LLM do th
        #In parameters part we will specify all those things that will help LLM to get parameters from the prompt
        #also thorugh this parameter information LLM will figure out whether to use the specifies function or not
        "parameters": {
            #parameter will be of type object
            "type": "object",
            #---------------------------------------------
            #our parameter will have following properties
            "properties": {
                "location": {
                    "type": "string",

                    #this desciption is also very important because it will be passed directly to language model used
                    #as we will pass location for the wheather so we will give description according to it
                    "description": "The city and state, e.g. San Francisco, CA",
                },
                #secondly temperature unit
                "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]},
            },
            #---------------------------------------------

            #the required parameter to hit the function is location (we use required parameter to make LLM understand that
            "required": ["location"],
        },
        #************************************************************
    }
]
```

In [83]:
```
#message that will be passed to LLM
messages = [
    {
        "role": "user",
        "content": "What's the weather like in Boston?"
    }
]
```

In [84]:
```
import openai
```

In [85]:
```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo-0613",
    messages=messages,
    #as we are specifying funcitons here
    functions=functions
)
```

In [86]:
```
print(response)
```


```
{
  "id": "chatcmpl-8UULl6QxdSFbAfCxTt5aHiHL48ObM",
  "object": "chat.completion",
  "created": 1702277197,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": null,
        "function_call": {
          "name": "get_current_weather",
          "arguments": "{\n  \"location\": \"Boston, MA\",\n  \"unit\": \"celsius\"\n}"
        }
      },
      "finish_reason": "function_call"
    }
  ],
  "usage": {
    "prompt_tokens": 82,
    "completion_tokens": 26,
    "total_tokens": 108
  },
  "system_fingerprint": null
}
```


The Response will have null as content (In content we basically have the repsonse of LLM), but as it hits the function specified by us it have information in function_call object which will be dictionary and it will have:

```
-name of the functions that was hit for the prompt
-arguements that were taken from the prompt
```

Response of the Prompt:

In [87]:
```
response_message = response["choices"][0]["message"]
```

In [88]:
```
response_message
```


```
<OpenAIObject at 0x7fce643af0e0> JSON: {
  "role": "assistant",
  "content": null,
  "function_call": {
    "name": "get_current_weather",
    "arguments": "{\n  \"location\": \"Boston, MA\",\n  \"unit\": \"celsius\"\n}"
  }
}
```


Content will be Null in this Case because function was hit:

In [89]:
```
response_message["content"]
```

So the function call aruguement have the information:

In [90]:
```
response_message["function_call"]
```

```
<OpenAIObject at 0x7fce27d099a0> JSON: {
  "name": "get_current_weather",
  "arguments": "{\n  \"location\": \"Boston, MA\",\n  \"unit\": \"celsius\"\n}"
}
```

Arguements that LLM got from the Prompt are:

In [91]:
```
json.loads(response_message["function_call"]["arguments"])
```

```
{'location': 'Boston, MA', 'unit': 'celsius'}
```

In [92]:
```
args = json.loads(response_message["function_call"]["arguments"])
```

Openai does not call the function by itself which is made by us but instead give us the name of the function and paramters that we can use to call the function and then we can explicitly call the function:

Calling the Function by using the arguements that LLM got for us:

In [93]:
```
get_current_weather(args)
```

```
'{"location": {"location": "Boston, MA", "unit": "celsius"}, "temperature": "72", "unit": "fahrenheit", "forecast": ["sunny",
"windy"]}'
```

It used the function to reutrn the same hard coded value of temperautre.

Now lets give it a prompt that is not related to function and lets see what does LLM do with that:

In [94]:
```
messages = [
    {
        "role": "user",
        #as hi message is not related to function that is specified for the wheather
        "content": "hi!",
    }
]
```

In [95]:
```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo-0613",
    messages=messages,
    functions=functions,
)
```

In [96]:
```
print(response)
```

```
{
  "id": "chatcmpl-8UULsO22kFrMDlsCUAkdqwYW26WBU",
  "object": "chat.completion",
  "created": 1702277204,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello! How can I assist you today?"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 76,
    "completion_tokens": 10,
    "total_tokens": 86
  },
  "system_fingerprint": null
}
```

Now as you can see above LLM demonstrated that prompt is not related to function, so response does not have the functions call part and instead LLM replied in the content part by taking it as a normal Prompt

1. There is a parameter that we can use to specify that whether Language model should choose on its own to use function (get parameters for the function) or not, or should we force it to consider function for every type of Prompt:

In [97]:
```
    {
        "role": "user",
        "content": "hi!",
    }
]
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo-0613",
    messages=messages,
    functions=functions,
    #so it is set to auto that means we are giving freedom to LLM to choose on the basis of the Prompt (that is default cas
    function_call="auto",
)
print(response)
```


```
{
  "id": "chatcmpl-8UULuVG2wNNwiJR11IpYfeZZ7RZMY",
  "object": "chat.completion",
  "created": 1702277206,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello! How can I assist you today?"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 76,
    "completion_tokens": 10,
    "total_tokens": 86
  },
  "system_fingerprint": null
}
```


2. Second type of value we can use for function_call is none according to which we are forcing LLM to not consider any function:

In [98]:
```
    {
        "role": "user",
        #giving the prompt that does not need any function
        "content": "hi!",
    }
]
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo-0613",
    messages=messages,
    functions=functions,
    function_call="none",
)
print(response)
```


```
{
  "id": "chatcmpl-8UULvxxaloUCpQFDE6rxkw12iLIfT",
  "object": "chat.completion",
  "created": 1702277207,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello! How can I assist you today?"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 77,
    "completion_tokens": 9,
    "total_tokens": 86
  },
  "system_fingerprint": null
}
```


Now lets give type of input that is compatible with function use, but we are forcing LLM to not consider any function:

In [99]:
```
        #now according to the message it should tell us to call the get current wheather function
        "content": "What's the weather in Boston?",
    }
]
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo-0613",
    messages=messages,
    functions=functions,
    #now lets set it to none, that means do not consider any of the funcitons provided
    function_call="none",
)
print(response)
```


```
{
  "id": "chatcmpl-8UULw3yWvrgsvG7PDVh4QyH08KeT7",
  "object": "chat.completion",
  "created": 1702277208,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Please wait a moment while I fetch the current weather in Boston."
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 82,
    "completion_tokens": 13,
    "total_tokens": 95
  },
  "system_fingerprint": null
}
```


As you can see above according to the message it should tell us to call get current wheather function but according to function_call paramter which is specifie to none that means we are telling LLM to not consider any function, It does not returned the function and returned the simple response

3. The Final type of thing we can do with function_call parameter is by forcing the LLM to consider the Function:

In [100]:
```
        "role": "user",
        #prompt that is general means not compatible with function use
        "content": "hi!",
    }
]
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo-0613",
    messages=messages,
    functions=functions,
    #forcing LLM to consider the function named get_current_wheather
    function_call={"name": "get_current_weather"},
)
print(response)
```


```
{
  "id": "chatcmpl-8UULwTRd1ZabbzbpZWiG5r6YhjOss",
  "object": "chat.completion",
  "created": 1702277208,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": null,
        "function_call": {
          "name": "get_current_weather",
          "arguments": "{\n  \"location\": \"San Francisco, CA\"\n}"
        }
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 83,
    "completion_tokens": 12,
    "total_tokens": 95
  },
  "system_fingerprint": null
}
```

It does use the Function but the Arugements that it returned are completely wrong

Now lets give it the Prompt Compatible with Function and force it use the function:

In [101]:
```
messages = [
    {
        "role": "user",
        "content": "What's the weather like in Boston!",
    }
]
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo-0613",
    messages=messages,
    functions=functions,
    function_call={"name": "get_current_weather"},
)
print(response)
```


```
{
  "id": "chatcmpl-8UULxc0qUVbEseEn1SWoaeoQRSdPa",
  "object": "chat.completion",
  "created": 1702277209,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": null,
        "function_call": {
          "name": "get_current_weather",
          "arguments": "{\n  \"location\": \"Boston, MA\"\n}"
        }
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 89,
    "completion_tokens": 11,
    "total_tokens": 100
  },
  "system_fingerprint": null
}
```


Succesfully got the arguements and instruct us to use the function.

Now we will see how to feedback the response of the function to LLM back:

In [102]:
```
response['choices'][0]['message']
```


```
<OpenAIObject at 0x7fce28427400> JSON: {
  "role": "assistant",
  "content": null,
  "function_call": {
    "name": "get_current_weather",
    "arguments": "{\n  \"location\": \"Boston, MA\"\n}"
  }
}
```


In [103]:
```
messages.append(response["choices"][0]["message"])
```

In [104]:
```
messages
```


```
[{'role': 'user', 'content': "What's the weather like in Boston!"},
 <OpenAIObject at 0x7fce28427400> JSON: {
   "role": "assistant",
   "content": null,
   "function_call": {
     "name": "get_current_weather",
     "arguments": "{\n  \"location\": \"Boston, MA\"\n}"
   }
 }]
```


So we appended the response of the LLM in the message.

In [105]:
```
response
```


```
<OpenAIObject chat.completion id=chatcmpl-8UULxc0qUVbEseEn1SWoaeoQRSdPa at 0x7fce27d09590> JSON: {
  "id": "chatcmpl-8UULxc0qUVbEseEn1SWoaeoQRSdPa",
  "object": "chat.completion",
  "created": 1702277209,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": null,
        "function_call": {
          "name": "get_current_weather",
          "arguments": "{\n  \"location\": \"Boston, MA\"\n}"
        }
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 89,
    "completion_tokens": 11,
    "total_tokens": 100
  },
  "system_fingerprint": null
}
```


In [106]:
```
args = json.loads(response["choices"][0]["message"]['function_call']['arguments'])
#calling the function using the arguements
observation = get_current_weather(args)
```

Appending the result we got from the function in messages as role as function:

In [109]:
```
messages.append(
        {
            "role": "function",
            "name": "get_current_weather",
            "content": observation,
        }
)
```

In [110]:
```
messages
```


```
[{'role': 'user', 'content': "What's the weather like in Boston!"},
 <OpenAIObject at 0x7fce28427400> JSON: {
   "role": "assistant",
   "content": null,
   "function_call": {
     "name": "get_current_weather",
     "arguments": "{\n  \"location\": \"Boston, MA\"\n}"
   }
 },
 {'role': 'function',
  'name': 'get_current_weather',
  'content': '{"location": {"location": "Boston, MA"}, "temperature": "72", "unit": "fahrenheit", "forecast": ["sunny", "wind
y"]}'},
 {'role': 'function',
  'name': 'get_current_weather',
  'content': '{"location": {"location": "Boston, MA"}, "temperature": "72", "unit": "fahrenheit", "forecast": ["sunny", "wind
y"]}'}]
```


Passing back to LLM:

In [111]:
```
response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo-0613",
    messages=messages,
)
print(response)
```


```
{
  "id": "chatcmpl-8UUNlI0jz6PhTIiatO7dgeUT4BroG",
  "object": "chat.completion",
  "created": 1702277321,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "The current weather in Boston is sunny and windy with a temperature of 72\u00b0F."
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 118,
    "completion_tokens": 17,
    "total_tokens": 135
  },
  "system_fingerprint": null
}
```


In [ ]: