Analytics Vidhya

# Levels of Agents Complexity


Instructor

**Lucas Soares**

AI Engineer at Otovo
Specialist in LLM Applications & Computer Vision
Instructor at O'Reilly Media
Technical Writer & Content Creator

# *Agents in 3 Levels of Complexity*

Analytics Vidhya

# Level 1: LLM + functions inside the prompt

Inspired by 'Toolformer'

Link: https://arxiv.org/pdf/2302.04761.pdf


Analytics Vidhya

# Level 1: LLM + functions inside the prompt

```
from openai import OpenAI
client = OpenAI()

def get_response(prompt_question, model="gpt-3.5-turbo-16k"):
    response = client.chat.completions.create(
        model=model,
        messages=[{"role": "system", "content": "You are a helpful research and pr
                {"role": "user", "content": prompt_question}]
    )

    return response.choices[0].message.content

def create_directory(directory_name):
    subprocess.run(["mkdir", directory_name])

def create_file(file_name):
    subprocess.run(["touch", file_name])

def list_files():
    subprocess.run(["ls"])
```

# Level 1: LLM + functions inside the prompt

```
task_description = "Create a folder called 'lucas-the-agent-master'. Inside that
output = get_response(f"""Given this task: {task_description}, \n
                        Consider you have access to the following functions:

def create_directory(directory_name):
    '''Function that creates a directory given a directory name.'''
    subprocess.run(["mkdir", directory_name])

def create_file(file_name):
    '''Function that creates a file given a file name.'''
    subprocess.run(["touch", file_name])

def list_files():
  '''Function that lists all files in the current directory.'''
    subprocess.run(["ls"])

Your output should be the first function to be executed to complete the task con
The OUTPUT SHOULD ONLY BE THE PYTHON FUNCTION CALL and NOTHING ELSE.
""")

Markdown(output)

# Output:
# create_directory('lucas-the-agent-master')
```

# Level 1: LLM + functions inside the prompt

- Now, all we need is to find a way to execute this function.
- We can use Python's built in exec method for that:

```
exec("model." + output)
!ls -d */ | grep lucas
# Output:
# lucas-the-agent-master/
```

# Limitations

- Probabilistic outputs make function calls unreliable
- Need for structured ways to prepare the inputs of the function calls
- Putting entire functions inside text prompts is clunky and non-scalable

*Solution?* OpenAI Funtions!

# OpenAI's Function Calling API

# OpenAI Function Calling

- OpenAI function calling: standard way to connect models to outside tools.

  Link: https://platform.openai.com/docs/guides/function-calling

# OpenAI Function Calling

## Steps

1. Call the model with the user query and a set of functions defined in the function's parameter.
2. The model can choose to call one or more functions; if so, the content will be a stringified JSON object adhering to your custom schema.
3. Parse the string into JSON in your code, and call your function with the provided arguments if they exist.
4. Call the model again by appending the function response as a new message, and let the model summarize the results back to the user.

# Step 1 & 2

```
import json

def create_directory(directory_name):
    # Function to create a directory
    subprocess.run(["mkdir", directory_name])
    return json.dumps({"directory_name": directory_name})

tool_create_directory = {
    "type": "function",
    "function": {
        "name": "create_directory",
        "description": "Create a directory given a directory name.",
        "parameters": {
            "type": "object",
            "properties": {
                "directory_name": {
                    "type": "string",
                    "description": "The name of the directory to create.",
                }
            },
            "required": ["directory_name"],
        },
    },
}
tools = [tool_create_directory]
```

# Step 1 & 2

```
def run_terminal_task():
    messages = [{"role": "user", "content":
    "Create a folder called 'lucas-the-agent-master'."}]
    tools = [tool_create_directory]
    response = client.chat.completions.create(
        model="gpt-3.5-turbo-16k",
        messages=messages,
        tools=tools,
        tool_choice="auto",
    )
    response_message = response.choices[0].message
    tool_calls = response_message.tool_calls
    # Check if the model called a function
    if tool_calls:
        # Proceed to step 3
```

# Step 3: Parse and execute the function

```
  available_functions = {
    "create_directory": create_directory,
}
messages.append(response_message)
for tool_call in tool_calls:
    function_name = tool_call.function.name
    function_to_call = available_functions[function_name]
    function_args = json.loads(tool_call.function.arguments)
    function_response = function_to_call(
        directory_name=function_args.get("directory_name"),
    )
    messages.append(
        {
            "tool_call_id": tool_call.id,
            "role": "tool",
            "name": function_name,
            "content": function_response,
        }
    )
```

# Step 4: Summarize Results Back to User

```
  second_response = client.chat.completions.create(
    model="gpt-3.5-turbo-16k",
    messages=messages,
)
return second_response

output = run_terminal_task()
```

# Thank You

Analytics Vidhya