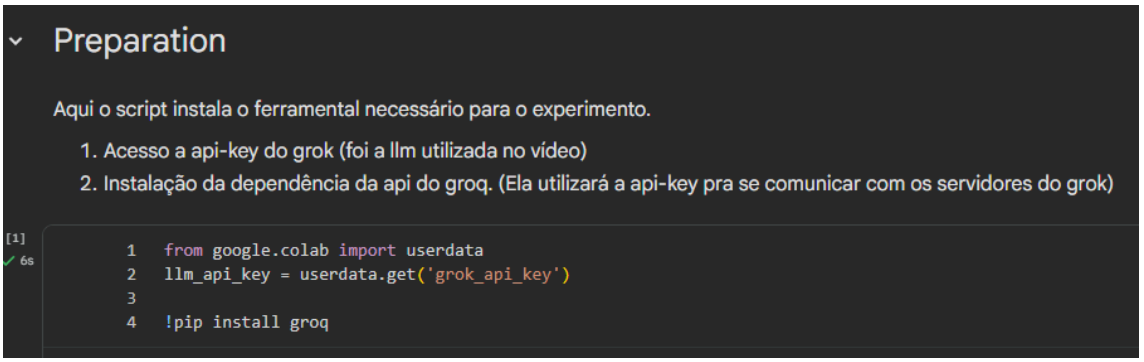
Relatório 2 - Prática: Python: Criando um ReAct Agent do Zero (I)

Danillo Guimarães

Descrição da atividade

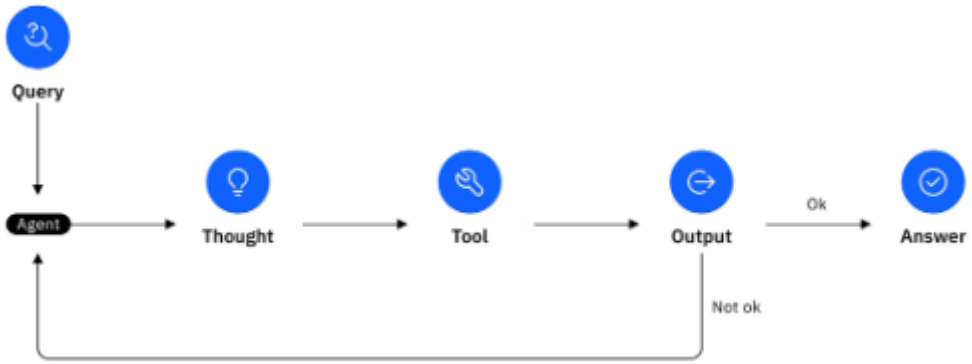
A idéia do agente ReAct fez todo sentido, com o agente conduzindo as interações, e tomando as decisões. Como não utilizamos nenhum ferramental (como podemos observar na imagem abaixo) pra tomar decisões ou esconder qualquer passo, ficou claro o procedimento de raciocínio e de delegação.



Também pode ser conferido no teste de configuração, onde inicilizamos o groq com a api, e realizamos uma interação.

O material da IBM (O que é um agente ReAct) fornece informações adicionais ao tutorial do card. Deixa claro que ReAct é um framework básico e fundamental para desenvolvimento de agentes e que existem outros. Na parte mais técnica ele afirma o papel da engenharia de

prompt na estruturação das atividades. O artigo representa em imagem o loop do agente



A customização mais profunda com relação a abordagem foi a mudança da estrutura do agente, onde internalizei o função do loop.

```python
import re

class Agent:

  def __init__(self, client, system):
    self.client = client
    self.system = system
    self.messages = []
    if self.system is not None:
      self.messages.append({"role": "system", "content": self.system})


  def __call__(self, message = ""):
    if message is not None:
      self.messages.append({"role": "user", "content": message})

    result = self.execute()

    self.messages.append({"role": "assistant", "content": result})
    return result


  def execute(self):
    completion = self.client.chat.completions.create(
      messages=self.messages,
      model="llama-3.3-70b-versatile",
    )
    return completion.choices[0].message.content


  def loop(self, max_iterations=10, query: str = ""):

    available_tools = {
        "get_menu": get_menu,
        "calculate": calculate
    }

    i = 0

    next_prompt = query

    while i < max_iterations:
        i += 1
        result = self.__call__(next_prompt)
        print("")
        print(f"--- Iteração {i} ---")
        print(result)

        if "PAUSE" in result and "Action" in result:
            # REGRA REGEX:
            # 1. ([a-z_]+) -> Nome da ferramenta
            # 2. (?::\s*(.+))? -> Opcional. Procura ':' seguido de espaço e o
            argumento
            match = re.search(r"Action: ([a-z_]+)(?::\s*(.+))?", result, re.
            IGNORECASE)

            if match:
                chosen_tool = match.group(1)
                arg = match.group(2)

                if chosen_tool in available_tools:

                    if chosen_tool == "get_menu":
                      result_tool = available_tools[chosen_tool]()

                      # Ou chamando por:
                      # result_tool = get_menu()

                    else:
                      result_tool = available_tools[chosen_tool](arg)

                      # Ou chamando por:
                      # result_tool = calculate(arg)

                    next_prompt = f"Observation: {result_tool}"
                else:
                    next_prompt = "Observation: Tool not found"

                print(next_prompt)
                continue
```

Dificuldades

A maior dificuldade foi com o python, pois ainda estou aprendendo. Como tentava implementar pelo colab e não tem o apoio de uma ide robusta, eu perdia muito tempo. Tive dificuldades nas expressões regulares, e na chamada das tools, pois as minhas tinham assinaturas diferentes - uma tinha argumento, a outra não.
Dentro da área específica da definição dos agentes, enfrentei dificuldades pra identificar as restrições do agente que eu incluiria no system prompt. Por exemplo, se eu permitisse escolher várias pizzas, a função de cálculo deveria ser melhor definida. Quanto mais eu me aproximo da interação no mundo real, mais restrições eu precisaria de colocar o prompt, o que humanamente não escala, pois além da atualização do system prompt, preciso capacitar o agente com ferramentas/tools.

Conclusões

O ReAct é um framework de codificação de agentes que permite uma experiencia de agentes. Contém suas limitações, mas é basilar para o entendimento da dinâmica e entendimento de outros frameworks. .

Referencias

- https://www.youtube.com/watch?v=hKVhRA9kfeM
- https://www.ibm.com/br-pt/think/topics/react-agent