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

Kaíque Medeiros Lima

Introdução

Esse relatório descreve a implementação e execução de um agente de chat em Python, baseado no ciclo *Thought–Action–PAUSE–Observation*. O objetivo foi demonstrar, de forma prática, como o agente formula um pensamento, seleciona uma ação (ferramenta interna), aguarda uma observação e, ao final, retorna uma resposta conclusiva.

Descrição da Atividade

O que são Agentes ReAct?

Agentes ReAct são sistemas que operam em ciclos de raciocínio, ação e pausa, permitindo uma interação dinâmica com o ambiente. Eles utilizam ferramentas internas para realizar tarefas específicas, como cálculos ou consultas a bancos de dados, e são projetados para lidar com incertezas e variabilidades nas respostas.

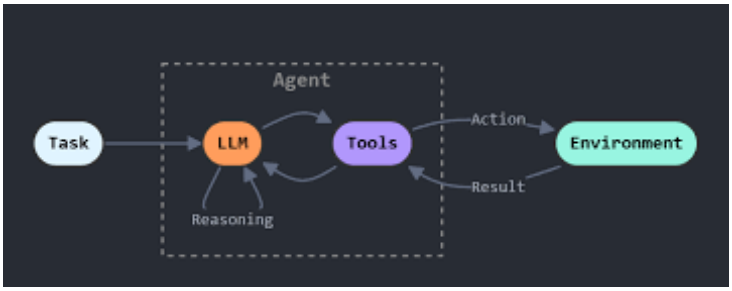

Figura 1: Medium - Demystifying Agents: ReAct-Style Agents vs "Agentic Workflows"

Implementação do Agente (Aula)

Primeiro, define-se uma chave de API e inicializa-se o cliente Groq para chamadas de chat. Em seguida, estabelece-se um *system prompt* que orienta o agente a operar em loops cíclicos de pensamento, ação e pausa.

Na sequência, cria-se a classe `Agent`, responsável por:

- Armazenar as mensagens de sistema, usuário e assistente.

- Enviar requisições de chat à API (modelo `llama3-70b-8192`).
- Interpretar a resposta, extraindo comandos de cálculo ou consulta de massa planetária.

Em um exemplo prático, o agente recebeu a pergunta *"What is the mass of Earth times 5?"*. Ele executou:

1. **Thought**: Identificou que precisava obter a massa da Terra.
2. **Action**: Chamou a ferramenta `get_planet_mass` com o argumento "Earth".
3. **PAUSE**: Aguardou a observação (massa retornada: 5.972e24 kg).
4. **Thought**: A partir da observação, concluiu que deveria multiplicar por 5.
5. **Action**: Chamou `calculate` com a operação "5.972e24 5".
6. **PAUSE**: Aguardou o resultado da multiplicação.
7. **Answer**: Apresentou o valor final 2.9860e25 kg.

Na minha opinião, essa estrutura garante clareza entre cada etapa de raciocínio e facilita o rastreamento de falhas.

Implementação do Agente (Prática)

A implementação do agente no código prático foi feita utilizando a mesma lógica que o agente da aula, mas com adaptações de conteúdo da IA. A implementação foi para um agente que calcula a quantidade de proteína de três diferentes fontes, sendo elas: Peito de frango, carne de boi e ovos. O agente em questão foi capaz de calcular a quantidade de proteína em 300g de peito de frango. A principal mudança foi a adaptação do regex para extrair o comando de cálculo e a adaptação do prompt inicial para refletir o contexto de nutrição.

Dificuldades

Alguns pontos exigem atenção especial:

- A atualização correta do histórico de mensagens entre iterações, para evitar repetições indesejadas.
- A sincronização entre Observação e próximo passo; observar o valor correto antes de prosseguir é crucial.
- A adaptação do regex para a extração de comandos, garantindo que o agente interprete corretamente as respostas do modelo.

Conclusão

Na minha opinião, o agente demonstra de forma eficiente como integrar ferramentas internas e um modelo de linguagem para realizar cálculos e consultas estruturadas. Contudo, a robustez depende da consistência do padrão de saída e do tratamento de erros nas ações.

Referências

[1] Groq Inc. *Groq API*, consultado em Maio de 2025.

[2] Medium. *Demystifying Agents: ReAct-Style Agents vs "Agentic Workflows"*, consultado em Maio de 2025.