Laboratório 24: Agente Meteorológico com LangChain

Objetivo do Laboratório

O objetivo desta atividade é implementar o padrão ReAct e *Function Calling* utilizando o framework **LangChain**. Construiremos um Agente Autônomo capaz de usar ferramentas customizadas, interceptar o planejamento do LLM e interagir com uma API real (ou função local) de clima, resolvendo o problema da falta de atualização temporal dos modelos.

Atividade Prática

1. Configuração do Ambiente

Abra o repositório da disciplina e instale as dependências essenciais do LangChain:

```
pip install langchain langchain-openai python-dotenv
```

Crie o arquivo `agente_clima.py`.

2. Criando as Ferramentas (Tools)

No LangChain, as ferramentas são funções normais do Python anotadas com `@tool`. A *Docstring* é vital, pois é ela que será enviada para o LLM entender quando e como usar essa ferramenta.

```
import os
from dotenv import load_dotenv
from langchain.agents import tool
from langchain_openai import ChatOpenAI
from langchain.agents import create_tool_calling_agent, AgentExecutor
from langchain_core.prompts import ChatPromptTemplate

load_dotenv()
```

```
@tool
def get_weather(location: str) -> str:
    """Retorna a temperatura e clima atuais para a cidade especificada."""
    # Em um projeto real, fariamos um requests.get("api.weather...")
    # Aqui, simularemos para simplificar
    temperaturas = {
        "São Paulo": "22°C, Nublado",
        "Rio de Janeiro": "30°C, Ensolarado",
        "Curitiba": "15°C, Chuvoso"
    }
    return temperaturas.get(location, "Temperatura desconhecida para esta cidade.")

# Agrupando as ferramentas
tools = [get_weather]
```

3. O Loop do Agente e a Execução

Precisamos montar o prompt e instanciar o `AgentExecutor`.

```
# 1. Configurar o LLM
llm = ChatOpenAI(model="gpt-3.5-turbo", temperature=0)

# 2. Prompt Padrão
prompt = ChatPromptTemplate.from_messages([
    ("system", "Você é um assistente prestativo. Use as ferramentas se necessário."),
    ("human", "{input}"),
    ("placeholder", "{agent_scratchpad}"),
])

# 3. Criar a rotina do Agente e o Executor (Loop ReAct)
agent = create_tool_calling_agent(llm, tools, prompt)

agent_executor = AgentExecutor(
    agent=agent,
    tools=tools,
    verbose=True # Importante para vermos o log de raciocínio (Thought/Action)
)

# 4. Invocação
print("Iniciando Agente...")
pergunta = "Qual é o clima agora no Rio de Janeiro?"
resposta = agent_executor.invoke({"input": pergunta})

print("\n=== Resposta Final ===")
print(resposta["output"])
```

4. Entrega via Git

1 Execute o arquivo. Observe as linhas destacadas em verde no console (*verbose=True*). O modelo dirá: "Preciso chamar a ferramenta `get_weather` com 'Rio de Janeiro'". A função será rodada e retornará "30°C, Ensolarado". O modelo então gera a resposta para o usuário humano.

2 Faça o commit no repositório: `git add agente_clima.py` e `git commit -m "Adiciona Agente Autônomo com uso de Tools no LangChain"`.

3 Envie para o servidor remoto (GitLab).

Critério de Avaliação

O laboratório está concluído quando for evidenciado (por vídeo rápido do terminal ou verificação do log do console) a presença do raciocínio interceptador do LangChain (o fluxo Thought → Action → Observation → Answer) conectando o modelo de texto ao dicionário de temperatura do Python.