Xây dựng "Planner Agent" – Trợ lý lập kế hoạch tìm kiếm thông minh

Mục tiêu

Tạo một agent có khả năng **phân tích câu hỏi đầu vào** và đề xuất **3 truy vấn tìm kiếm web** hợp lý để nghiên cứu sâu về chủ đề đó. Agent này sẽ **không tìm kiếm** mà chỉ đưa ra **kế hoạch tìm kiếm**.

Bước 1: Cài đặt thư viện cần thiết

```
pip install openai pydantic
```

Bước 2: Định nghĩa cấu trúc đầu ra

Sử dụng Pydantic để định nghĩa schema cho dữ liệu mà mô hình sẽ trả về.

```
from pydantic import BaseModel, Field
from typing import List

class WebSearchItem(BaseModel):
    reason: str = Field(..., description="Lý do tại sao nên thực hiện truy vấn này.")
    query: str = Field(..., description="Cụm từ khóa cần tìm kiếm trên internet.")

class WebSearchPlan(BaseModel):
    searches: List[WebSearchItem] = Field(..., description="Danh sách các tìm kiếm để trả lời câu hỏi.")
```

Bước 3: Tạo prompt hướng dẫn hệ thống

```
SYSTEM_PROMPT = """
Bạn là một trợ lý nghiên cứu hữu ích.
```

```
Dựa trên một câu hỏi đầu vào, hãy đề xuất 3 truy vấn tìm kiếm trên web tốt nhất để trả lời câu hỏi đó.
Mỗi truy vấn cần bao gồm:
1. Lý do thực hiện truy vấn đó.
2. Cụm từ khóa cụ thể cần tìm kiếm.

Trả lời dưới dạng một đối tượng JSON có dạng:
{
  "searches": [
    {"reason": "...", "query": "..."},
    {"reason": "...", "query": "..."},
    {"reason": "...", "query": "..."}
  ]
}
"""
```

Bước 4: Gọi mô hình GPT để lập kế hoạch tìm kiếm

```
import openai
import json

openai.api_key = "YOUR_OPENAI_API_KEY"  # Thay bằng API key của bạn

def generate_search_plan(question: str):
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[
            {"role": "system", "content": SYSTEM_PROMPT},
            {"role": "user", "content": f"Câu hỏi: {question}"}
        ],
        temperature=0.7
    )

    content = response["choices"][0]["message"]["content"]

    try:
        # Chuyển JSON thành đối tượng WebSearchPlan
        data = json.loads(content)
        return WebSearchPlan(**data)
    except Exception as e:
        print("Lỗi khi phân tích phản hồi:", e)
```

```
        print("Phản hồi thô:", content)

# Thử nghiệm
plan = generate_search_plan("Các framework AI phổ biến trong năm 2025 là gì?")
if plan:
    for idx, item in enumerate(plan.searches, start=1):
        print(f"\n Tìm kiếm #{idx}")
        print(f"Lý do : {item.reason}")
        print(f"Truy vấn: {item.query}")
```

Kết quả mong đợi

```
{
  "searches": [
    {
      "reason": "Tìm hiểu các framework AI mới được giới thiệu năm 2025",
      "query": "framework AI mới năm 2025"
    },
    {
      "reason": "Phân tích các xu hướng ứng dụng AI trong ngành công nghiệp",
      "query": "ứng dụng AI trong công nghiệp 2025"
    },
    {
      "reason": "Tìm các công bố tại hội nghị AI hàng đầu năm 2025",
      "query": "hội nghị AI công bố framework 2025"
    }
  ]
}
```

Ghi nhớ

- Đầu ra có cấu trúc giúp mô hình **lập luận rõ ràng hơn** (chain of thought).
- Khi yêu cầu mô hình giải thích "lý do" trước khi trả lời, bạn sẽ nhận được **câu trả lời chất lượng hơn**.
- Tất cả đầu ra thực chất là **JSON**, không có gì "ma thuật" cả.

Tác giả: Đỗ Ngọc Tú

Công Ty Phần Mềm VHTSoft

Phiên bản #1
Được tạo 2 tháng 5 2025 08:34:44 bởi Đỗ Ngọc Tú
Được cập nhật 2 tháng 5 2025 08:38:33 bởi Đỗ Ngọc Tú