Career Agent

This project features a career agent built using **LangChain** and **LangGraph**. The agent automates the job search and application process by performing several key tasks: parsing a user's resume, searching for relevant jobs, matching the user's skills to job descriptions, and generating a personalized cover letter.

The agent's workflow is powered by a state machine, which allows it to move through a sequence of nodes, with each node representing a specific step in the job search process.

□ Architecture

The core of the agent is a **LangGraph state machine**. The state is a list of `BaseMessage` objects, which allows for a conversational history to be maintained throughout the process. This history is crucial as later nodes can reference the output of earlier nodes (e.g., the `cover_letter_node` uses the output of the `profile_node` and `matched_node`).

The graph consists of four main nodes, each performing a distinct function:

- **`profile_node`** : Extracts key information such as skills, roles, and years of experience from a user's resume. This is the starting point of the workflow.
- **`job_search_node`** : Takes the extracted profile information and uses it to formulate a precise search query for an external search tool (Tavily). It then fetches and summarizes job listings.
- **`matched_node`** : Analyzes the fetched job listings and the user's profile to rank and recommend the top three most suitable jobs.
- **`cover_letter_node`** : Generates a personalized cover letter for the best-matched job, leveraging the user's profile and the job description to create a tailored and compelling message.

The graph is designed as a linear sequence: `profile` ➡ `search` ➡ `matched` ➡ `cover` ➡ `END`

□ How It Works

1. **Resume Ingestion**: The agent starts with a `HumanMessage` containing the user's resume or career information.
2. **Profile Parsing**: The `profile_node` uses an LLM to parse the raw text and structure the user's skills and experience.
3. **Job Search**: The `job_search_node` constructs an optimal search query and uses the `tavily` tool to find relevant jobs.
4. **Job Matching**: The `matched_node` uses another LLM call to compare the user's profile with the search results and identify the best fits.
5. **Cover Letter Generation**: The `cover_letter_node` generates a draft cover letter, specifically tailored to the top-ranked job and the user's unique background.

The final output is a list of messages, with the last message containing the generated cover letter.

□ Requirements

- **LangChain**: The framework for building the agent.
- **LangGraph**: The library for orchestrating the stateful, multi-step workflow.
- **Tavily**: A search API for finding job listings. You will need to set your API key as an environment variable (`SEARCH_API_KEY`).
- **A Large Language Model (LLM)**: An API from providers like OpenAI, Anthropic, or Google to power the reasoning and generation steps. The code uses `model.invoke()` , so you'll need to configure your model of choice.

⚙ Usage

To run the agent, you must first set up your API keys and the LLM provider.

1. **Set up API keys**:

   ```
    export SEARCH_API_KEY="YOUR_TAVILY_API_KEY"
   export OPENAI_API_KEY="YOUR_OPENAI_API_KEY"
   # or other LLM keys
   ```

2. **Run the script**:

   ```
   python career_agent.py
   ```

   The script will execute the example workflow and print the output, which includes the parsed profile, job search results, ranked jobs, and the final cover letter.

□ Example Output

The script prints each step of the agent's reasoning and generation process, showcasing the agent's ability to act autonomously.

```
 HumanMessage: search a job for me, i am a startup builder...
AIMessage: extracted profile:
skill: python, prompt engineer, langchain, ai agent, RAG, chatbot, multi-agent system
roles: startup builder, data management
year of experience: 1 year
...
AIMessage: jobs found:
... (summarized job listings from Tavily)
...
AIMessage: Based on your profile and the job listings provided, here are the top 3 recommended jobs for you:
1. AI Agent Developer at [Company A]: This job is a perfect match given your skills in building AI agents, RAG, and multi-agent syst
2. Python Developer (AI focus) at [Company B]: This role directly leverages your Python, LangChain, and AI agent development skills.
3. Prompt Engineer at [Company C]: Your experience in prompt engineering and building custom chatbots is highly relevant here. This
...
AIMessage: [A personalized cover letter for the top-ranked job]
```