UC Berkeley MOOC Hackathon 2024: Building a Flight Agent with Gemini (Powered by Tavily API)

✈️🛩️🤖

This past weekend, I participated in the UC Berkeley MOOC Hackathon 2024. My project, powered by the Tavily API and Gemini, aimed to create a flight agent capable of providing real-time flight information and answering user queries. This post details my journey, the challenges faced, and the future potential of this exciting project.

What is an AI Agent?

At its core, an AI agent is a software program designed to autonomously perform tasks by interacting with its environment. It simulates intelligent behavior, using predetermined rules or trained models to make decisions in real-time. Think of it as a digital assistant with a specific purpose and the ability to learn and adapt. It operates within a loop, receiving input, processing it, taking action, observing the results, and refining its approach.

The Flight Agent: How it Works

My flight agent follows this core agent loop. It receives a user query, utilizes a Large Language Model (LLM), in this case Gemini, and takes action by querying the Tavily API. The Tavily API is a powerful web search engine that allows the agent to retrieve relevant flight information. The agent then observes the results and presents the final response to the user.

The workflow is structured as follows:

1. **Receive data:** The agent receives the user's query.
2. **Analyze data:** The LLM contextualizes and interprets the query.
3. **Decide on action:** The agent determines the best course of action, which involves querying the Tavily API with relevant parameters.
4. **Act:** The agent executes the action and retrieves flight information.
5. **Output:** The agent presents the retrieved information to the user.

Types of AI Agents

- **ReAct**
- **Multi Agent**
- **Self Reflecting**
- **Planning**

There are various types of AI agents, each with its unique capabilities. My flight agent falls under the category of a *ReAct (Reasoning and Acting)* agent. It dynamically combines reasoning and action within a loop, adjusting its approach based on the information it gathers.

Benefits of AI Agents

The advantages of AI agents are numerous:

- **Increased efficiency:** Automating repetitive tasks boosts productivity.
- **Better decision-making:** AI agents can spot patterns, trends, and correlations that humans might miss.
- **Improved customer experience:** Personalized interactions and proactive assistance enhance user satisfaction.
- **Cost savings:** Automating processes reduces the need for manual labor and resources.

Challenges and Risks

Building AI agents comes with its own set of challenges:

- **Lack of transparency:** Understanding the decision-making process of learning agents can be complex.
- **Data bias:** Agents rely on data, and biased data can lead to unfair or discriminatory results.
- **Ethical considerations:** Programming moral decision-making into agents is a complex issue.
- **Security risks:** Agents are susceptible to cyberattacks, potentially leading to data leaks or compromised functionality.

What's the future of AI Agents?

The rise of AI agents has marked a significant shift in how we approach work. The future of the AI world is not about humans versus AI, but rather humans working alongside AI.

Besides risk associated with AI Agents ,I foresee strong potential in their abilities.

Addressing these challenges proactively will be essential to harnessing the full power of AI agents for the benefit of humanity.

We can expect this in future from agents:

- Increased autonomy and sophistication
- Enhanced Human AI collaboration
- Prioritizing Ethical AI
- Improved Personalization and Customization
- Specialized Agents for Specific Tasks

Building the Flight Agent: Tools and Frameworks

I leveraged several frameworks for this project, including Langchain for agent development and the Tavily API for flight data retrieval.

The Code: A Glimpse Under the Hood

The code utilizes several key functions:

- get_api_session(): Establishes an authorized session with the Tavily API.
- fetch_flight_data(): Retrieves flight data based on the flight ID and API session.
- get_flight_status(): Decorated with @tool, this function retrieves flight status information and serves as a tool for the agent.

The agent itself is initialized using initialize_agent(), specifying the tools, LLM, agent type, and verbosity.

Future Enhancements

I envision expanding the flight agent's capabilities to include real-time flight tracking, integration with other travel-related services, and more sophisticated prompt engineering techniques. This would allow the agent to answer more complex queries about layovers, delays, gate changes, and other travel-related information. Direct integration with airline APIs is also a future goal.

Conclusion

Building this flight agent was a rewarding experience. It highlighted the potential of AI agents to revolutionize how we interact with information and services. While challenges remain, the future of AI agents is bright, promising more seamless and personalized experiences across various domains. I'm excited to continue developing this project and exploring the vast possibilities of this technology.