

5 AI Agent Projects To Try

ReAct Agent

(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

Question

(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.

Act 1: `Search[Apple Remote]`

Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the `Front Row media center` program ...

Thought

Action

Observation

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search `Front Row` next and find what other device can control it.

Act 2: `Search[Front Row]`

Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]

Thought 3: Front Row is not found. I need to search `Front Row (software)` .

Act 3: `Search[Front Row (software)]`

Obs 3: Front Row is a discontinued media center software ...

Thought 4: `Front Row (software)` is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.

Act 4: `Finish[keyboard function keys]`



From "*REACT: SYNERGIZING REASONING AND ACTING IN LANGUAGE MODELS*" (Shunyu et al., 2022)

Free Course Inside

ReAct Search Agent

The modern ReAct (Reason + Act) Search Agent has replaced the Simple Reflex Agent concept, making it more suitable for decision-making in complex environments. ReAct agents can combine search capabilities with dynamic reasoning, and tools like LangGraph, AutoGen, or CrewAI can help streamline the process.

In this project, you will design a ReAct Search Agent capable of solving dynamic search problems, such as answering complex questions from a web database, retrieving and organizing relevant information, or planning a route based on real-time data.

Technologies Used

- LangGraph, AutoGen, or CrewAI frameworks for building ReAct agents.
- Search Tools like Serper will be used to retrieve information from google or other tools for search.
- LLM integration to enable reasoning and natural language processing.



An Autonomous Flight Simulation Agent

The goal of the Agent Pilot project is to train a deep learning model to fly a simulated aircraft with no human assistance. This AI needs to co-ordinate many parameters including altitude, speed, weather and fuel while at the same time meeting flight safety procedures and regulation. When applying the reinforcement learning, the agent starts solving problems by taking decisions according to the environment – for instance, deviation from storms, optimization of fuel consumption, or level(choice) to decrease turbulence.

The same as the flight control the implements for the creation of the flight simulator can be either general-use implemented FlightGear or a customized built one in Python using the Pygame. The AI has to work with several variables from the sensors (altitude, speed and distance to other objects) and apply control adjustments.

Technologies Used

- Reinforcement Learning for teaching the agent to make optimal flight decisions.
- Simulated Environments using tools like FlightGear or OpenAI Gym for flight simulation.
- Sensor Data Integration to interpret the environment (altitude, weather, etc.).



Autonomous HR Agent

The Autonomous HR Agent project involves automating key HR processes like job application screening, resume parsing, candidate ranking, and initial interviews. By integrating Large Language Models (LLMs) and function calling, this agent goes beyond traditional rule-based systems. It can now parse resumes using Natural Language Processing (NLP), extract relevant details (skills, experience, education), match them against job descriptions, and even initiate dynamic function calls to schedule interviews or rank candidates.

The agent can conduct the initial interview stages using LLM-based conversational AI, enabling it to pose HR-specific questions, interpret candidate responses, and evaluate their suitability. This agent can use sentiment analysis and context-aware AI to adjust interview questions dynamically.

Technologies Used

- LLMs and Function Calling to automate recruitment decisions.
- NLP for resume analysis and parsing
- Machine Learning for candidate ranking and scoring.
- Automation Tools for seamless integration into HR workflows.



Content Recommendation Agent

The Content Recommendation Agent is designed to provide personalized recommendations based on users' interactions, such as browsing history, queries, or click behavior. By leveraging LLMs and reinforcement learning, the agent can offer highly tailored content suggestions. LLMs enhance the Natural Language Understanding (NLU) component, enabling more accurate matching of content to user preferences.

The agent can combine collaborative filtering and content-based filtering with LLM-powered contextual understanding to recommend articles, products, or media that align with the user's needs. As the agent gathers more user data, reinforcement learning allows it to refine its recommendations over time.

Technologies Used

- LLMs for advanced natural language understanding and personalized content suggestions.
- Collaborative Filtering Algorithms to make recommendations based on user preferences.
- Content-Based Filtering to recommend similar content based on item properties (e.g., video topics, product categories).
- Data Analytics for tracking user behavior and improving recommendation accuracy.



AI Agent for Game Development

The purpose of this project is that an AI sensitive should be created that can learn from environment through play experience in the typed of video games. Reinforcement learning is also a type of learning that depends on system update; the agent will be trained to get better in the game, to become familiar with the environment and respond depending upon the results being a reward or punishment. This can be done beginning with basic number guessing game or tic tac toe and up to games like chess or the one created as a platformer.

The agent will incorporate the Q-learning techniques or the Deep Q-Networks (DQNs) to enhance the performance of its actions in the gaming arena. This way, specific past moves will permit the agent to determine whether it should start attacking an opponent or, on the contrary, avoid a trap.

Technologies Used

- Reinforcement Learning for teaching the agent to improve its gameplay.
- Python Game Development Libraries like Pygame to create or interface with game environments.
- Game Theory and AI Decision-Making for strategy optimization.