Autonomous Agents - Architectures & Challenges

Abi Aryan
AI Agent Discussion #1
04th Aug 2023

Agent Architecture

Conceptual designs that determine

- how AI agents obtain precepts from the environment,
- how they process this information, and
- how they generate actions to interact with their environment

Chains vs Agents

Agents go beyond Chains by incorporating an LLM-based decision-making framework and an autonomous process. They tackle tasks with flexibility, choosing tools and their order strategically.

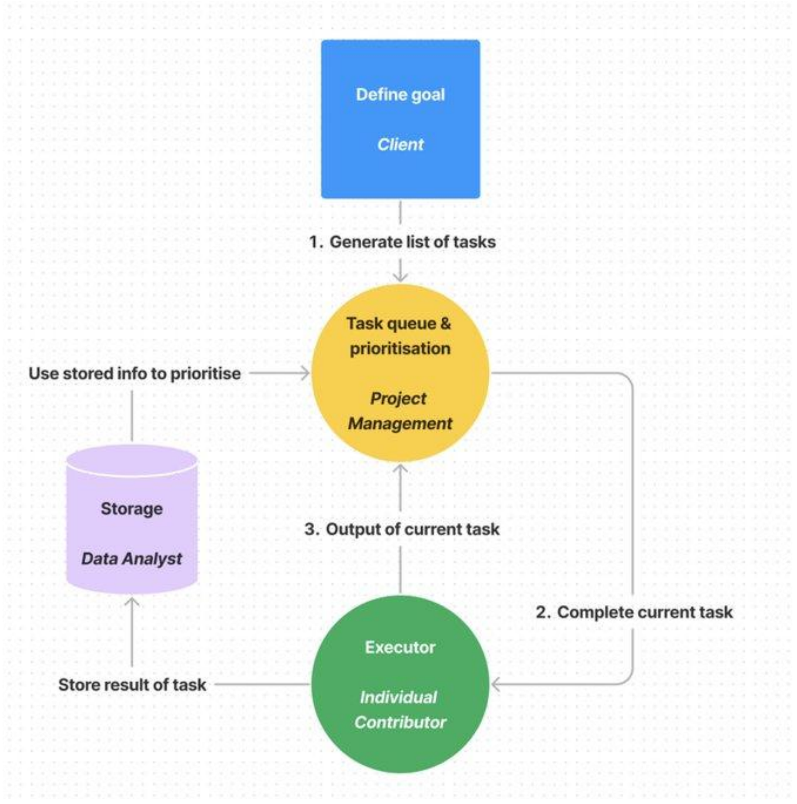
A single Agent may complete a mission solo, or it can interact with other Agents. Chaining Agents is one way to go, but obviously, Agents can interact in many different ways, setups, and hierarchies

LangChainAI - Building applications with LLMs, Chains, and Agents through composability. Includes two types of Agents:

- “Action Agents”: decide-act-observe-repeat, one action at a time
- “Plan-and-Execute Agents”: plan-act-observe-repeat, a set of actions at a time

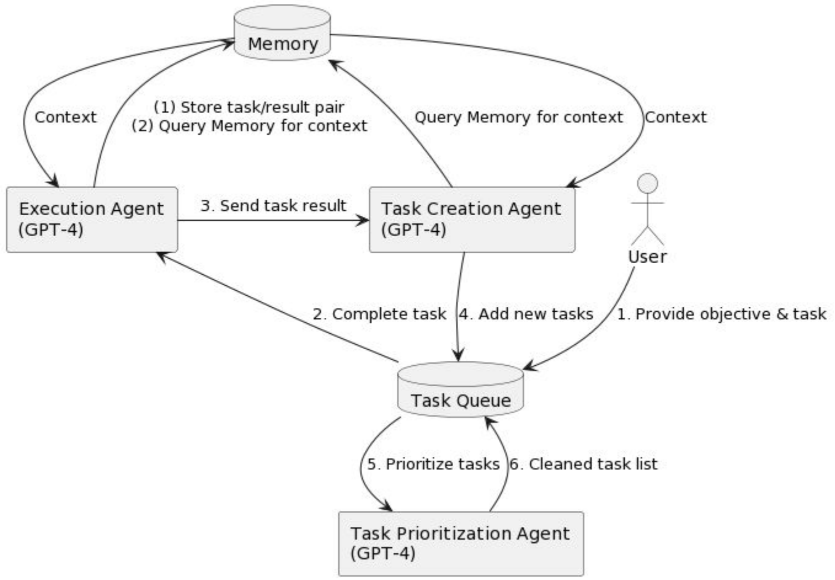
Action Agents: The agent learns to call external APIs for extra information that is missing from the model weights (often hard to change after pre-training), including current information, code execution capability, access to proprietary information sources and more.

"Plan-and-Execute Agents"



How any autonomous agent works

1. Initialize Goal: Define the objective for the AI.
2. Task Creation: The AI checks its memory for the last X tasks completed (if any), and then uses it's objective, and the context of it's recently completed tasks, to generate a list of new tasks.
3. Task Execution: The AI executes the tasks autonomously.
4. Memory Storage: The task and executed results are stored in a vector database.
5. Feedback Collection: The AI collects feedback on the completed task, either in the form external data or internal dialogue from the AI. This feedback will be used to inform the next iteration of the Adaptive Process Loop.
6. New Task Generation: The AI generates new tasks based on the collected feedback and internal dialogue.
7. Task Prioritization: The AI reprioritizes the task list by reviewing it's objective and looking at the last task completed.
8. Task Selection: The AI selects the top task from the prioritized list, and proceeds to execute them as described in step 3.
9. Iteration: The AI repeats steps 4 through 8 in a continuous loop, allowing the system to adapt and evolve based on new information, feedback, and changing requirements.



Westworld, Camel, BabyAGI, AutoGPT

1) Agent Creation

2) Memory and Retrieval

3) Reflection (ReAct, Reflexion, CoH)

4) Planning (CoT, ToT)

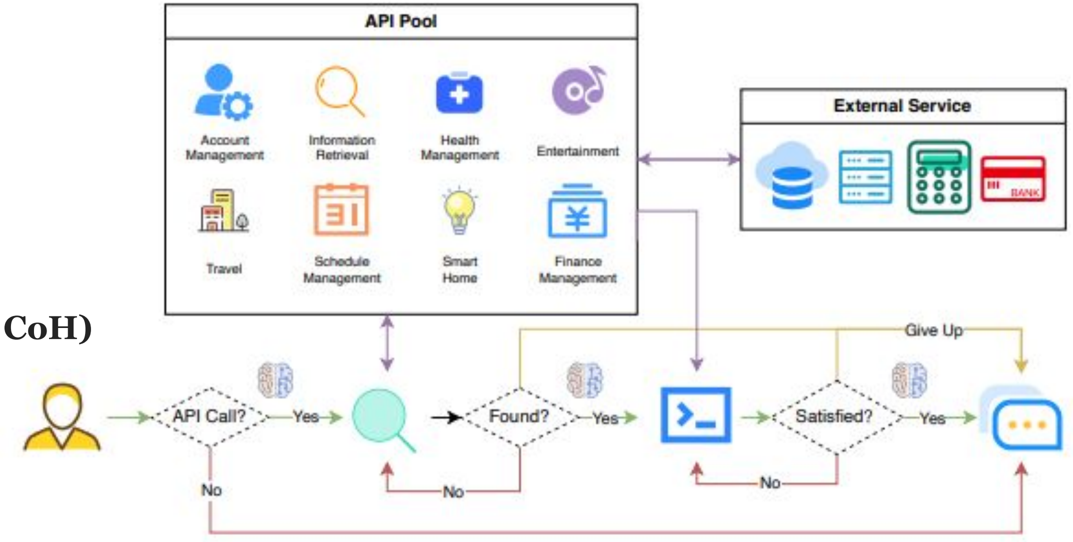
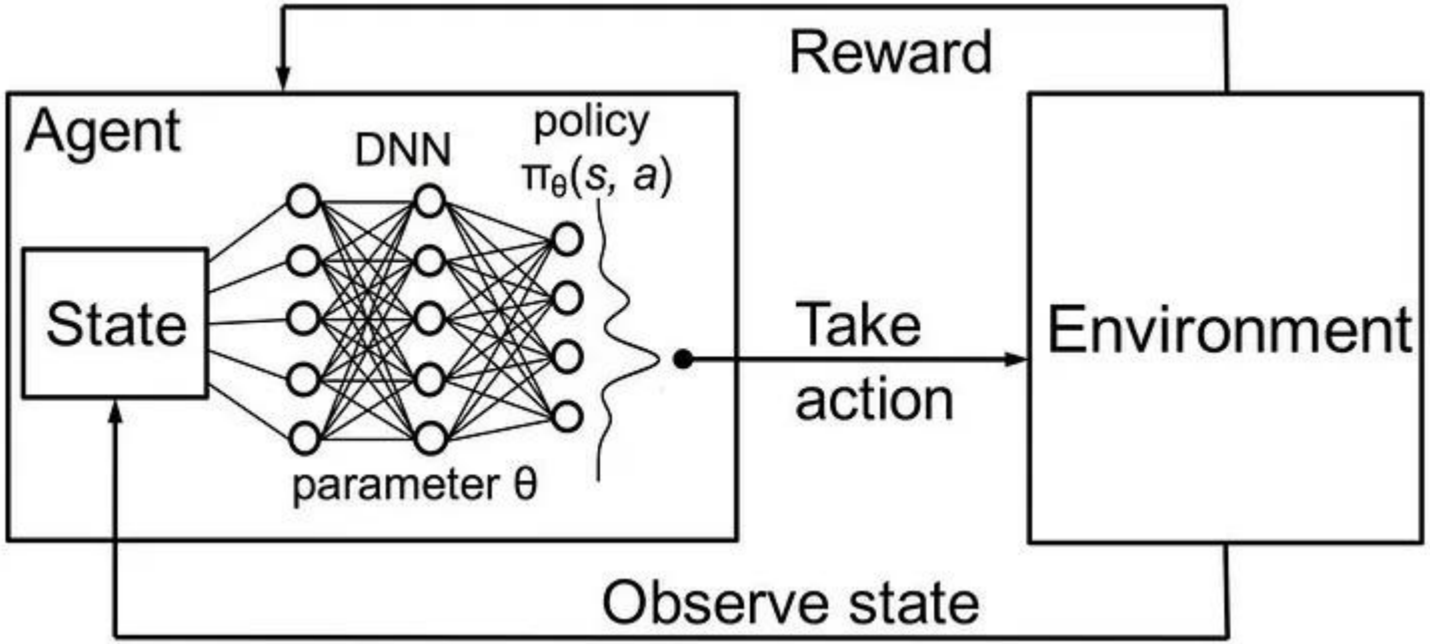


Figure 1: The proposed Tool-Augmented LLMs paradigm.

But that's a limited picture

- **Blackboard Architecture:** In this architecture, agents communicate and collaborate by reading from and writing to a common global data structure, often referred to as the "blackboard". This architecture can be useful in problem-solving where agents contribute different knowledge and skills.
- **Contract Net Protocol:** This is a task-sharing protocol where one agent (the manager) proposes a task that other agents (contractors) can bid on. The manager then assigns the task to the agent with the best bid. This architecture is commonly used in distributed problem solving.
- **Peer-to-Peer Architecture:** Here, each agent can communicate directly with any other agent. There is no central controlling agent, and all agents are equal. This architecture is typically used in distributed systems where agents have equal authority and capability.
- **Master-Slave Architecture:** This architecture features a central controlling agent (master) and multiple subordinate agents (slaves). The master delegates tasks to slaves and collects results.
- **Broker Architecture:** A broker agent is responsible for routing communication between other agents, ensuring that requests from a client agent are satisfied by a server agent. This architecture can help reduce the complexity of direct communication between many agents.
- **Team-Based Architecture:** This architecture is often used when a group of agents needs to work collaboratively to solve problems. Agents in a team share common goals and work cooperatively to achieve them.
- **Hierarchical Architecture:** In this structure, agents are organized in a hierarchical manner, with higher-level agents directing the actions of lower-level agents. This can be useful for complex tasks that can be decomposed into subtasks.
- **Multi-Agent Reinforcement Learning (MARL) Architectures:** In some MAS, the agents learn their behavior based on feedback from the environment, and MARL algorithms are used to guide this learning. Some MARL architectures involve independent learning, where each agent learns its behavior independently of others. Others involve joint action learners, where agents learn policies based on the joint actions of all agents.

What's Missing: State Awareness



Depending on the architecture,

- **Auction Algorithms:** These include algorithms like the Vickrey–Clarke–Groves (VCG) auction and other mechanism design algorithms. They are used for resource allocation and task distribution among agents. Each agent bids for a task, and the task is assigned based on the bids and certain other rules.
- **Consensus Algorithms:** Consensus algorithms like the Paxos and Raft are used in MAS to agree on a single data value for a distributed network. These algorithms help in making the system robust against failures of some agents.
- **Swarm Intelligence Algorithms:** Inspired by the behavior of natural swarms (e.g., ants, bees, birds), these algorithms, like Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), are used for optimization and search problems in MAS.
- **Negotiation Algorithms:** These include algorithms for automated negotiation and bargaining among agents. For example, the Contract Net Protocol (CNP) is a classic negotiation protocol for task allocation in MAS.
- **Multi-Agent Reinforcement Learning (MARL) Algorithms:** These are adaptations of classic reinforcement learning algorithms (like Q-Learning and Deep Q-Networks) to multi-agent settings. They help the agents to learn from interactions with the environment and other agents to maximize some notion of cumulative reward.
- **Graph-based Algorithms:** These are used in situations where the multi-agent system can be modeled as a graph, such as the Dijkstra algorithm for shortest paths, or graph coloring algorithms for task assignment.
- **Distributed Constraint Optimization (DCOP) Algorithms:** In these algorithms, agents cooperate to find the best solution according to a global utility function that aggregates their individual utilities, under certain constraints. Algorithms like Max-Sum, DPOP, and ADOPT are used in DCOPs.
- **Game Theoretic Algorithms:** These algorithms use concepts from game theory, such as Nash equilibrium, to make decisions in competitive multi-agent scenarios.
- **Coordination Algorithms:** These include various methods for coordinating the actions of multiple agents, such as coordination graphs, Markov Decision Processes (MDPs), and Decentralized Partially Observable MDPs (Dec-POMDPs).
- **Distributed Search and Planning Algorithms:** These algorithms are used when multiple agents need to cooperate to solve a problem or create a plan. They include distributed versions of classic AI algorithms like A* and depth-first search.

Challenges

- Decision making, knowledge-intensive searches for answers and and complexity of mastering decision-making is non-trivial in novel environments
- A major challenge is to overcome common mistakes like repetitive action choice, cyclic hallucination, or random action choice.
- Updating the vector stores or knowledge bases and meta-data management

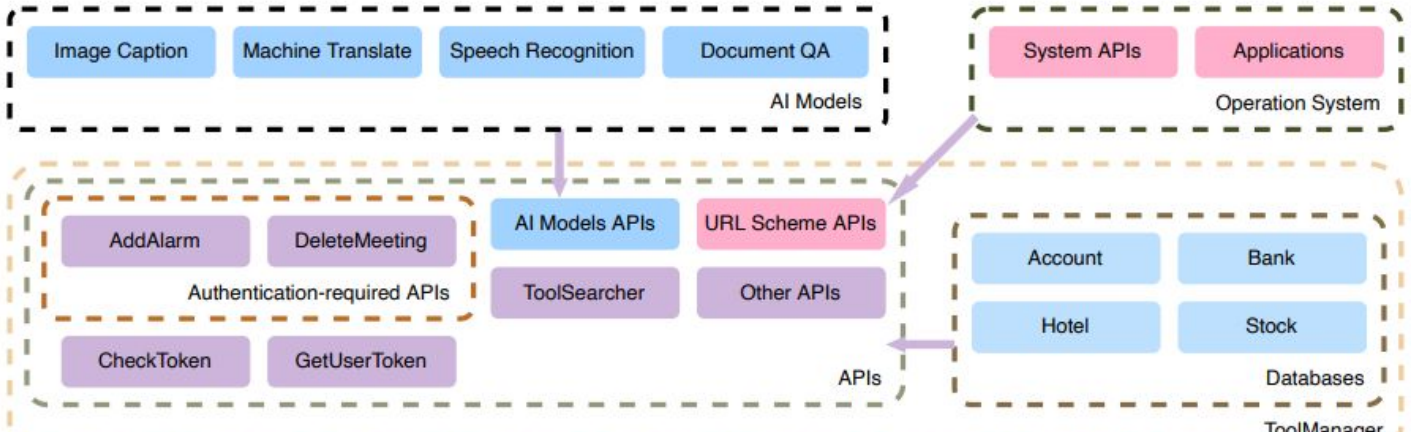


Image Caption
Machine Translate
Speech Recognition
Document QA
AI Models
System APIs
Applications
Operation System
AddAlarm
DeleteMeeting
Authentication-required APIs
AI Models APIs
URL Scheme APIs
ToolSearcher
Other APIs
CheckToken
GetUserToken
APIs
Account
Bank
Hotel
Stock
Databases
ToolManager