PLANK

AI WORKSHOP

Topics

- Introduction
- OpenAI
- LLMs Features
  - RAG
  - function callings
  - Vectorization
  - Fine-tuning
  - Text2sql
- LLM Agents - MCP
  - Key Features
- Hands-on: Creation of MCP Server

Introduction

- Introduction to Artificial Intelligence (AI) and its impact on various industries.
- AI simulates human intelligence through machine learning, logic, and perception
- Automates repetitive and decision-based tasks, reducing human error
- Used in predictive modeling, anomaly detection, and natural language processing (NLP)
- Enables innovation in fields like healthcare (diagnostics), finance (fraud detection), and law (contract analysis)

Introduction

The evolution of Large Language Models (LLMs).

- NLP models evolved from rule-based systems to data-driven deep learning architectures
- LLMs are trained on massive corpora (e.g. Common Crawl, Wikipedia, GitHub)
- They use transformer architectures (e.g. GPT, T5, BERT) to understand and generate language
- Models now support multilingual, multimodal, and memory-augmented capabilities

OpenAI: Overview including GPT models and research advancements.

- Founded in 2015 with the mission to ensure AGI (Artificial General Intelligence) benefits all of humanity
- Created state-of-the-art models like GPT-3, GPT-4, Codex, and DALL·E
- Pioneered public access to LLMs via ChatGPT, reaching 100M users in 2 months
- OpenAI APIs power thousands of applications across healthcare, finance, legal, and education
- Actively publishes research on alignment, interpretability, safety, and multi-modal models
- Introduced Function Calling, Assistants API, fine-tuning pipelines, and RAG-ready infrastructure

LLMs Features

Retrieval-Augmented Generation (RAG)

- Connects LLMs to external knowledge sources (e.g. web, databases, documents)
- Fetches up-to-date information before generating a response
- Enhances factual accuracy and contextual precision
- Common in tools like Perplexity, Bing Chat, and enterprise assistants

LLMs Features

Function Calling

- Allows LLMs to trigger external APIs or internal functions dynamically
- Enables actions like retrieving weather data, sending messages, or automating business processes
- Functions are described using structured JSON schemas
- Used in OpenAI GPT, LangChain, and agent frameworks

LLMs Features

Vectorization

- Represents text, images, or code as high-dimensional vectors (embeddings)
- Measures semantic similarity for search, clustering, or recommendations
- Enables vector databases (e.g. Pinecone, Weaviate, Qdrant)
- Key for similarity search and context retrieval in modern AI systems

LLMs Features

Fine Tuning

- Retrains a base model with custom, domain-specific data
- Increases model accuracy for specialized applications
- Supports supervised or instruction-based fine-tuning
- Examples: BloombergGPT, Med-PaLM, LegalMinds

LLMs Features

Text-to-SQL

- Translates natural language into SQL queries
- Empowers non-technical users to access databases and analytics
- Used in BI tools, data chatbots, and data democratization platforms
- Enables prompt-driven analytics

AI Agents

What is an AI agent?

AI agents are software systems that use AI to pursue goals and complete tasks on behalf of users. They show reasoning, planning, and memory and have a level of autonomy to make decisions, learn, and adapt.

Their capabilities are made possible in large part by the multimodal capacity of generative AI and AI foundation models. AI agents can process multimodal information like text, voice, video, audio, code, and more simultaneously; can converse, reason, learn, and make decisions. They can learn over time and facilitate transactions and business processes. Agents can work with other agents to coordinate and perform more complex workflows.

Source: https://cloud.google.com/discover/what-are-ai-agents?hl=e

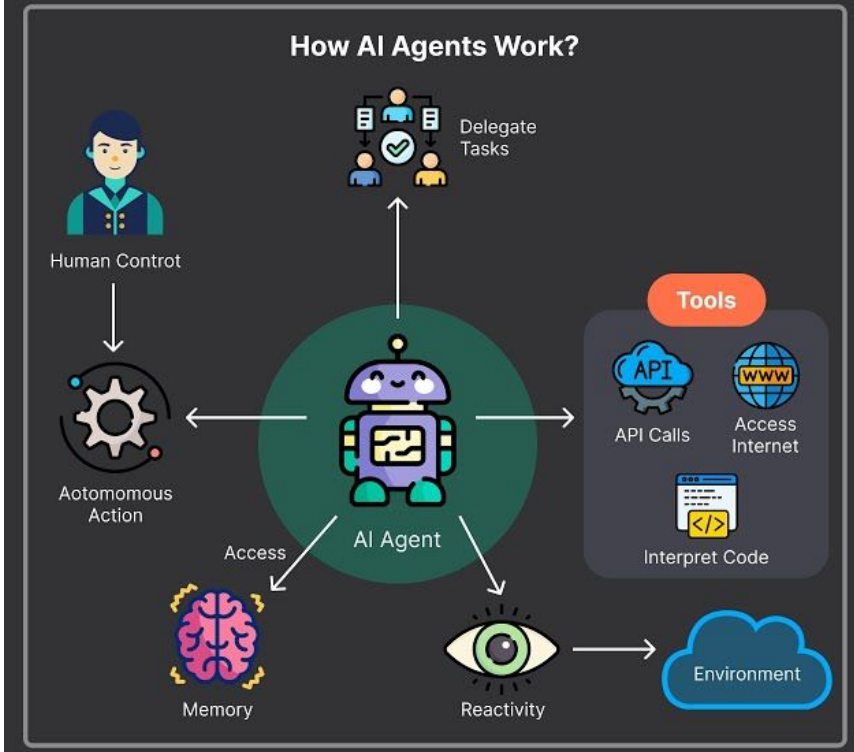
Key features of an AI agent

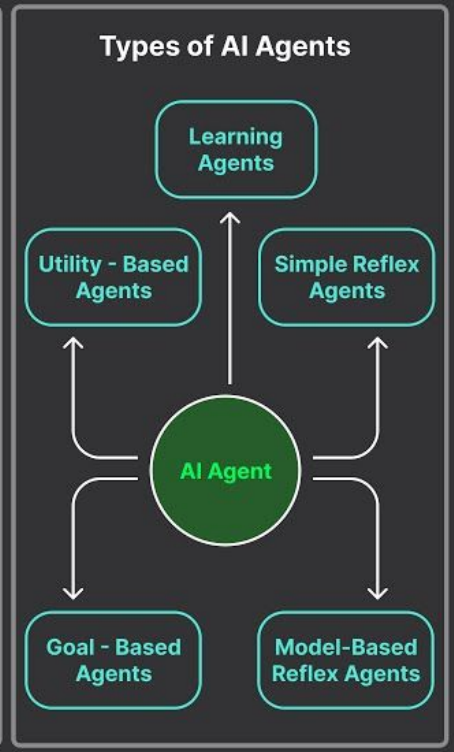
As explained above, while the key features of an AI agent are reasoning and acting (as described in ReAct Framework) more features have evolved over time.

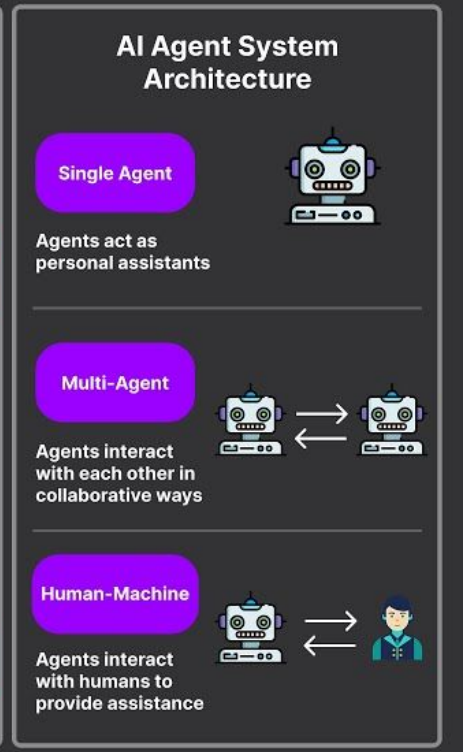
- **Reasoning**: This core cognitive process involves using logic and available information to draw conclusions, make inferences, and solve problems. AI agents with strong reasoning capabilities can analyze data, identify patterns, and make informed decisions based on evidence and context.
- **Acting**: The ability to take action or perform tasks based on decisions, plans, or external input is crucial for AI agents to interact with their environment and achieve goals. This can include physical actions in the case of embodied AI, or digital actions like sending messages, updating data, or triggering other processes.
- **Observing**: Gathering information about the environment or situation through perception or sensing is essential for AI agents to understand their context and make informed decisions. This can involve various forms of perception, such as computer vision, natural language processing, or sensor data analysis.
- **Planning**: Developing a strategic plan to achieve goals is a key aspect of intelligent behavior. AI agents with planning capabilities can identify the necessary steps, evaluate potential actions, and choose the best course of action based on available information and desired outcomes. This often involves anticipating future states and considering potential obstacles.
- **Collaborating**: Working effectively with others, whether humans or other AI agents, to achieve a common goal is increasingly important in complex and dynamic environments. Collaboration requires communication, coordination, and the ability to understand and respect the perspectives of others.
- **Self-refining**: The capacity for self-improvement and adaptation is a hallmark of advanced AI systems. AI agents with self-refining capabilities can learn from experience, adjust their behavior based on feedback, and continuously enhance their performance and capabilities over time. This can involve machine learning techniques, optimization algorithms, or other forms of self-modification.

Source: Source: https://cloud.google.com/discover/what-are-ai-agents?hl=en

What is an AI Agent ?







LLM Agents

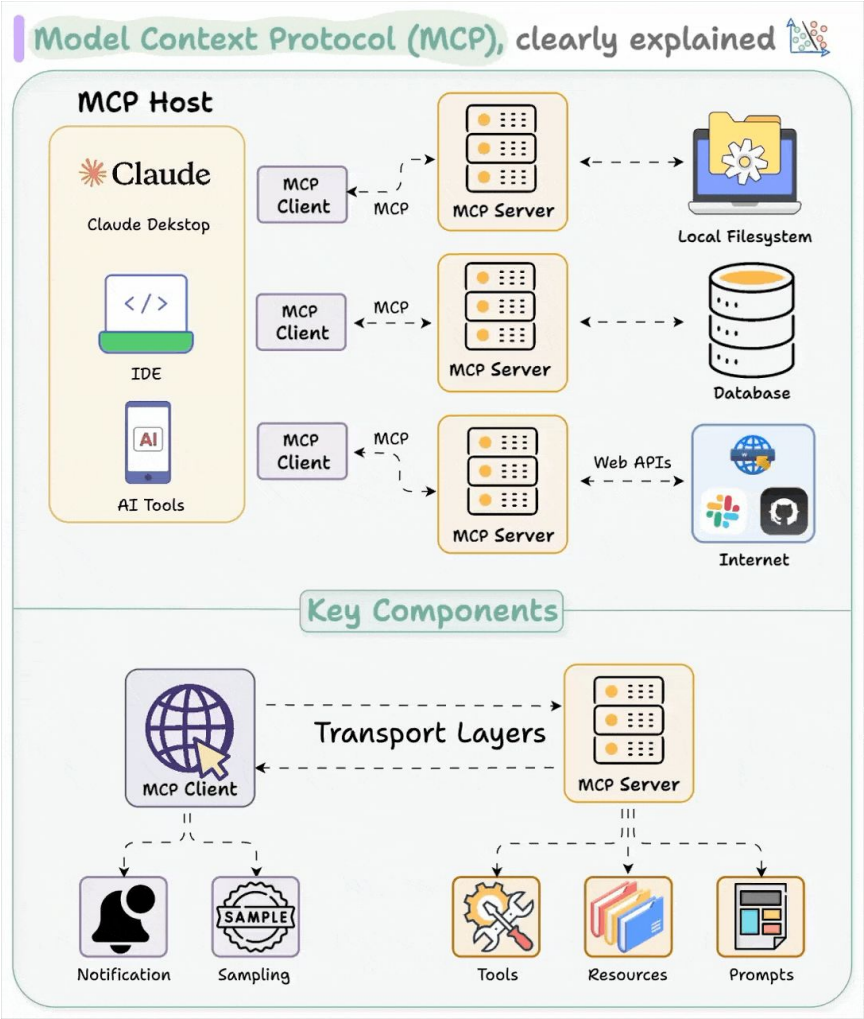
Model Context Protocol (MCP)

MCP is an open protocol that standardizes how applications provide context to LLMs. Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect your devices to various peripherals and accessories, MCP provides a standardized way to connect AI models to different data sources and tools.

Architecture:

- MCP Hosts: Programs like Claude Desktop, IDEs, or AI tools that want to access data through MCP
- MCP Clients: Protocol clients that maintain 1:1 connections with servers
- MCP Servers: Lightweight programs that each expose specific capabilities through the standardized Model Context Protocol
- Local Data Sources: Your computer's files, databases, and services that MCP servers can securely access
- Remote Services: External systems available over the internet (e.g., through APIs) that MCP servers can connect to

MCP


Model Context Protocol (MCP), clearly explained
MCP Host
Claude
Claude Dekstop
IDE
AI
AI Tools
MCP Client
MCP
MCP Server
Local Filesystem
Database
Web APIs
Internet
Key Components
Transport Layers
Notification
SAMPLE
Sampling
Tools
Resources
Prompts

Hands-On / Challenge

- Add 3 tools to the existing MCP Server
  - add_embedding_columns_to_table:
    - Parameters:
      - table_name: str,
      - embedding_column_name: str,
      - connection_string: str
    - What should do: should add a column with the suffix _embedding to a given table with this type: vector(1536)
  - generate_embedding:
    - Parameters:
      - column_name_from: str,
      - column_name_to: str,
      - table_name: str,
      - connection_string: str,
      - openai_api_key: str
    - What should do: Should select the records of a given *table_name*, use OpenAI API to generate the embedding using the model *text-embedding-ada-002* and then update the record using the *column_name_to* to persist the vector data
  - similarity_search:
    - Parameters:
      - query: str
      - table_name: str
      - embedding_column_name: str
      - connection_string: str
      - openai_api_key: str
      - match_threshold: float(between 0 and 1)
  - What should do: Should select the records of a given *table_name*, use OpenAI API to generate the embedding using the model *text-embedding-ada-002* and then update the record using the *column_name_to* to persist the vector data

Hands-On / Challenge

After finishing the hands-on exercises,

Please, zip the code and send it to

andre.gustavo@joinplank.com and

fernando.rocha@joinplank.com

The best submission will win a Plank gift plus a paid lunch with Plank specialists during a visit at Plank BH lab.