Kaggle | Google

5-Day Gen AI Intensive Course with Google 2025 Whitepaper Companion Podcast (Notes): Agents

Generative AI Agents: A Deep Dive

Introduction

- Humans use tools (books, search engines) to augment knowledge.
- Generative AI models can be trained to use tools for real-time information access and real-world actions.
    - Example: Using a database retrieval tool to access customer purchase history for tailored shopping recommendations.
    - Example: Making API calls to send emails or complete financial transactions.
- Agents combine reasoning, logic, and external information, extending beyond standalone Generative AI model capabilities.

What is a Generative AI Agent?

- An agent is a program that goes beyond a standalone generative AI model.
- Adds reasoning, logic, and access to external information.
- Achieves specific goals by:
    - Observing the environment.
    - Taking actions using available tools.
    - Operating independently once given a goal.
    - Figuring out necessary steps to achieve a goal without explicit instructions.
- Focus is on agents built using generative AI models for core decision-making.
- The language model acts as the "brain" of the operation.

Cognitive Architecture of Agents

- The cognitive architecture is the internal operating system of an agent, determining its behavior and decision-making process.
- Three main components:
    a. **Model:**
        - The language model itself, serving as the heart of the agent's intelligence.
        - Can be a single language model or multiple models working together.
        - Models can vary in size and specialization.
        - Utilizes instruction-based reasoning and logic frameworks (e.g., ReAct, Chain of Thought, Tree of Thoughts).
        - Models can be general-purpose, multimodal, or fine-tuned for specific tasks.
        - Flexibility allows choosing a model aligned with the agent's needs.
            - ♣ Example: Using a model trained on data similar to a specific API if the agent interacts with that API frequently.
        - Models are not typically pre-trained with specific agent configurations.
        - Fine-tuning can be done to provide examples of how the agent should use tools and reason through tasks.
    b. **Tools:**
        - Enable agents to perform actions in the real world.
        - Allow interaction with the physical world or other digital systems.
        - Provide the ability to reach beyond initial training data and affect change.
        - Varied complexity, often mapping to web APIs for information retrieval, data creation, modification, or deletion.
            - ♣ Example: Updating a customer record or fetching data from a website.
        - Key for Retrieval Augmented Generation (RAG).

        - Connect internal reasoning with the external world.

    c. **Orchestration Layer:**

        - The control center of the agent.
        - Manages information intake, reasoning, and decision-making.
        - Cycles through steps until the agent achieves its goal.
        - Ensures smooth and effective collaboration between the model and the tools.
        - Can be simple or complex, involving calculations, logic, or other machine learning algorithms.
        - Provides flexibility in controlling the agent.

Agents vs. Models

- Agents are more than just models with added tools.
- Key differences:

    a. **Knowledge:**

        - Models are limited to their training data.
        - Agents can access up-to-date external information, continuously learning.

    b. **Information Handling Over Time:**

        - Models make single predictions based on input.
        - Agents track interaction history, enabling multi-turn conversations with memory.

    c. **Tool Support:**

        - Models lack built-in support for tools.
        - Tools are integral to an agent's architecture.

    d. **Logic Layer:**

        - Models require prompt engineering to guide output.

        - Agents have a dedicated cognitive architecture incorporating reasoning frameworks.

How Agents Operate: Cognitive Architectures

- Analogy: Agent as a chef in a busy kitchen.
    - The chef follows a process: order received, ingredients assessed, dishes planned, actions taken (chopping, mixing, cooking), and adjustments made based on observations.
- Agents take in information, reason, make decisions, take actions, and learn from results.
- The orchestration layer tracks progress, manages the task state, and oversees reasoning and planning.
- Prompt engineering is crucial for guiding the agent's thought process.

Reasoning and Planning Frameworks

1. **ReAct (Reason and Act):**
    - Encourages step-by-step reasoning and interaction with the environment using tools.
    - Focuses on showing the user how the agent arrived at the answer.
    - Involves thinking about what needs to be done, trying an action, observing the result, and further reasoning based on the observation.
    - Leads to more reliable and transparent answers.
2. **Chain of Thought (CoT):**
    - Prompts the model to think through a problem in a structured, step-by-step manner.
    - Guides the model to explain its thought process, showing the logical steps to the solution.
    - Variations include self-consistency, active prompt, and multimodal CoT.
3. **Tree of Thoughts (ToT):**
    - An extension of CoT for problems requiring exploration of multiple possibilities.

- Branches out to explore multiple paths and anticipates responses.

ReAct in Practice: Example

- Scenario: User wants to book a flight.
- Agent prompts the model to generate the next step.
    a. Clarifying Question: "Where are you flying from?"
    b. Thought: Determine the best way to find flight information.
    c. Action: Call a flight API.
    d. Observation: Receive results from the API.
    e. Reasoning: Ask the user about preferred travel dates.
- The cycle repeats until all necessary information is gathered to book the flight.

Tools: Keys to the Outside World

- Language models process information and generate text but cannot act in the real world independently.
- Tools bridge the gap by connecting the model to external systems and data sources.
- Three main categories (for Google models):
    a. **Extensions:**
        - Standardized way to connect an agent to an API.
        - Simplify API usage without needing to manage low-level details.
        - The agent learns from examples of API usage.
        - Developed independently and included in the agent's configuration.
        - The agent dynamically chooses the most appropriate extension based on the user's query.
        - Example: Code interpreter (generates and executes Python code).

b. **Functions:**

- Self-contained pieces of code that perform specific tasks.
- The language model decides when to use each function and determines the arguments based on the function specification.
- The model decides which function to use and what data to provide, but it does not make the API call itself.
- Executed on the client-side.
- Provide more control, allowing handling of API calls as needed on the user's system.

c. **Data Stores:**

- Provide access to dynamic, up-to-date information.
- Agents can access external data sources like databases or APIs for current information.
- Allows access to internal company documents or research papers without retraining the model.
- Vector databases are commonly used for storing and searching information.
- Facilitates Retrieval Augmented Generation (RAG).

Retrieval Augmented Generation (RAG)

- Provides the language model with access to a vast external knowledge base.
- The model searches through data to find the most relevant information to answer a query.
- The external knowledge base can include web pages, PDFs, and spreadsheets.

Tools Recap

Tool	Execution Location	Best Used For
Extensions	Agent-side	Directly controlling API interactions, especially with pre-built extensions or when making multiple API calls.
Functions	Client-side	Security concerns, authentication issues, or fine-grained control over API calls.
Data Stores	Agent-side	Implementing RAG, providing access to wide-ranging external data sources.

Enhancing Model Performance

- Challenge: Ensuring the model chooses the correct tool.
- Approaches:
    a. **In-Context Learning:**
        - Providing the model with a specific prompt (recipe), relevant tools (ingredients), and examples of the desired output (finished dish).
    b. **Retrieval-Based In-Context Learning:**
        - Giving the model access to a vast pantry of ingredients and a library of cookbooks.
        - The model searches for the most relevant information and techniques.
        - Similar to how RAG works.
    c. **Fine-Tuning Based Learning:**
        - Specializing the model in a particular area.

        - Training the model on a large dataset of examples specific to the tasks and tools it should use.
- Combining approaches can yield the best results.

Agent Quick Start with LangChain

- Building a simple agent using LangChain and LangGraph.
- Uses the Gemini 2.0 fl001 model with tools for Google Search and Google Places API.
- Example Question: "Who did the Texas Longhorns play in their last game and what's the address of the stadium?"
    - Requires multiple steps: identify the opponent and find the stadium address.

Vertex AI Agents

- Vertex AI provides a managed platform for building, deploying, and managing agent applications.
- Includes user interfaces, evaluation tools, and systems for continuous improvement.
- Offers a natural language interface for defining agent behavior and tool usage.

Main Takeaways

- Agents are a major advancement from standalone language models.
- Agents can access real-time information, interact with the world, and plan/execute complex tasks.
- Orchestration layer is key, using cognitive architectures and reasoning techniques.
- Tools are essential for connecting the agent to the outside world.
- The future involves more sophisticated tools, improved reasoning, and agent chaining.
- Building agents is an iterative process of experimentation and refinement.