What is agentic reasoning?

What is agentic reasoning?

Agentic reasoning is a component of AI agents that handles decision-making. It allows artificial intelligence agents to conduct tasks autonomously by applying conditional logic or heuristics, relying on perception and memory, enabling it to pursue goals and optimize for the best possible outcome.

Earlier machine learning models followed a set of preprogrammed rules to arrive at a decision. Advances in AI have led to AI models with more evolved reasoning capabilities, but they still require human intervention to convert information into knowledge. Agentic reasoning takes it one step further, allowing AI agents to transform knowledge into action.

The "reasoning engine" powers the planning and tool calling phases of agentic workflows. Planning decomposes a task into more manageable reasoning, while tool calling helps inform an AI agent's decision through available tools. These tools can include application programming interfaces (APIs), external datasets and data sources such as knowledge graphs.

For businesses, agentic AI can further ground the reasoning process in evidence through retrieval-augmented generation (RAG). RAG systems can retrieve enterprise data and other relevant information that can be added to an AI agent's context for reasoning.

Your subscription will be delivered in English. You will find an unsubscribe link in every newsletter. You can manage your subscriptions or unsubscribe here. Refer to our IBM Privacy Statement for more information.

Agentic reasoning strategies

Agentic reasoning can be approached in different ways based on an agent's architecture and type. Here are some common techniques for AI agent reasoning, including the pros and cons of each:

- **Conditional logic**

- **Heuristics**
- **ReAct (Reason + Act)**
- **ReWOO (Reasoning WithOut Observation)**
- **Self-reflection**
- **Multiagent reasoning**

Conditional logic

Simple AI agents follow a set of preprogrammed condition-action rules. These rules usually take the form of “if-then” statements, where the “if” portion specifies the condition and the “then” portion indicates the action. When a condition is met, the agent carries out the corresponding action.

This reasoning methodology is especially suitable for domain-specific use cases. In finance, for instance, a fraud detection agent flags a transaction as fraudulent according to a set of criteria defined by a bank.

With conditional logic, agentic AI can’t act accordingly if it comes across a scenario it doesn’t recognize. To reduce this inflexibility, model-based agents use their memory and perception to store a current model or state of their environment. This state is updated as the agent receives new information. Model-based agents, however, are still bound by their condition-action rules.

For example, a robot navigates through a warehouse to stock a product on a shelf. It consults a model of the warehouse for the route it takes, but when it senses an obstacle, it can alter its path to avoid that obstacle and continue its traversal.

Heuristics

AI agent systems can also use heuristics for reasoning. Goal-based agents, for instance, have a preset goal. Using a search algorithm, they find sequences of actions that can help them achieve their goal and then plan these actions before conducting them.

For example, an autonomous vehicle can have a navigation agent whose objective is to suggest the quickest path to a destination in real-time. It can search through different routes and recommend the fastest 1.

Like goal-based agents, utility-based agents search for action sequences that achieve a goal, but they factor in utility as well. They employ a utility function to determine the most optimal outcome. In the navigation agent example, it can be tasked with finding not only the swiftest route but also 1 that will consume the least amount of fuel.

ReAct (Reason + Act)

This reasoning paradigm involves a think-act-observe loop for step-by-step problem-solving and iterative enhancement of responses. An agent is instructed to generate traces of its reasoning process,[1] much like what happens with chain-of-thought reasoning in generative AI (gen AI) models and large language models (LLMs). It then acts on that reasoning and observes its output,[2] updating its context with new reasoning based on its observations. The agent repeats the cycle until it arrives at an answer or solution.[2]

ReAct does well on natural language-specific tasks, and its traceability improves transparency. However, it can also generate the same reasoning and actions repeatedly, which can lead to infinite loops.[2]

ReWOO (Reasoning WithOut Observation)

Unlike ReAct, ReWOO removes the observation step and plans ahead instead. This agentic reasoning design pattern consists of 3 modules: planner, worker and solver.[3]

The planner module breaks down a task into subtasks and allocates each of them to a worker module. The worker incorporates tools used to substantiate each subtask with evidence and facts. Finally, the solver module synthesizes all the subtasks and their corresponding evidence to draw a conclusion.[3]

ReWOO outperforms ReAct on certain natural language processing (NLP) benchmarks. However, adding extra tools can degrade ReWOO's performance, and it doesn't do well in situations where it has limited context about its environment.[3]

Self-reflection

Agentic AI can also include self-reflection as part of assessing and refining its reasoning capabilities. An example of this is Language Agent Tree Search (LATS), which shares similarities with tree-of-thought reasoning in LLMs.

LATS was inspired by the Monte Carlo reinforcement learning method, with researchers adapting the Monte Carlo Tree Search for LLM-based agents.[4] LATS builds a decision tree that represents a state as a node and an edge as an action, searches the tree for potential action options and employs a state evaluator to choose a particular action.[2] It also applies a self-reflection reasoning step, incorporating its own observations as well as feedback from a language model to identify any errors in reasoning and recommend alternatives.[2] The reasoning errors and reflections are stored in memory, serving as additional context for future reference.[4]

LATS excels in more complex tasks such as coding and interactive question answering and in workflow automation, including web search and navigation.[4] However, a more involved approach and extra self-reflection step makes LATS more resource- and time-intensive compared to methods like ReAct.[2]

Multiagent reasoning

Multiagent systems consist of multiple AI agents working together to solve complex problems. Each agent specializes in a certain domain and can apply its own agentic reasoning strategy.

However, the decision-making process can vary based on the AI system's architecture. In a hierarchical or vertical ecosystem, 1 agent acts as a leader for AI orchestration and decides which action to take. Meanwhile, in a horizontal architecture, agents decide collectively.

Challenges in agentic reasoning

Reasoning is at the core of AI agents and can result in more powerful AI capabilities, but it also has its limitations. Here are some challenges in agentic reasoning:

- **Computational complexity**
- **Interpretability**
- **Scalability**

Computational complexity

Agentic reasoning can be difficult to implement. The process also requires significant time and computational power, especially when solving more complicated real-world problems. Enterprises must find ways to optimize their agentic reasoning strategies and be ready to invest in the necessary AI platforms and resources for development.

Interpretability

Agentic reasoning might lack explainability and transparency on how decisions were made. Various methods can help establish interpretability, and integrating AI ethics and human oversight within algorithmic development are critical to make sure agentic reasoning engines make decisions fairly, ethically and accurately.

Scalability

Agentic reasoning techniques are not 1-size-fits-all solutions, making it hard to scale them across AI applications. Businesses might need to tailor these reasoning design patterns for each of their use cases, which requires time and effort.