What is LangChain?

Tool calling overview

Tool calling refers to the ability of artificial intelligence (AI) models to interact with external tools, application programming interfaces (APIs) or systems to enhance their functions.

Instead of relying solely on pretrained knowledge, an AI system with tool-calling capabilities can query databases, fetch real-time information, execute functions or perform complex operations beyond its native capabilities.

Tool calling, sometimes referred to as function calling, is a key enabler of agentic AI. It allows autonomous systems to complete complex tasks by dynamically accessing and acting upon external resources.

Instead of just answering questions, large language models (LLMs) with tool calling can automate workflows, interact with databases, perform multistep problem-solving, make real-time decisions and more.

This shift is turning LLMs from passive assistants into proactive digital agents capable of carrying out complex tasks.

Why is tool calling important?

Large language models (LLMs) are traditionally limited by the data on which they are trained, a process that can be time and computationally intensive.

Even though leading LLMs are trained on vast datasets, the need for real-time data, external computations and enhanced interactivity led to the integration of tool calling capabilities.

Early LLMs, including OpenAI's GPT-2, were static. They generated responses based on their training data without the ability to fetch new information.

While impressive, they lacked real-world awareness and struggled with dynamic queries requiring live data, such as current events, stock prices or user-specific actions.

To address this limitation, developers began integrating external plug-ins, APIs and databases, allowing models to request and process real-time information rather than relying solely on static training data.

Developers trained LLMs to recognize when a query required external assistance. Moreover, external systems often have a particular input schema. Tool calling requests model responses that match the particular schema used by external systems.

How does tool calling work?

Tool calling involves several key components that work together to facilitate AI interaction with external tools. Modern LLMs including Anthropic's Claude, Meta's Llama 3, Mistral and IBM® Granite™ all possess tool calling capabilities but handle each a bit differently.

The first component is the AI model itself, which recognizes when it lacks sufficient knowledge or requires an external function to complete a request.

Next, the tool selection mechanism identifies the appropriate dependencies to handle the specific task, whether it is a search engine, a database or a computational resource.

When a tool is selected, the API interface comes into play, allowing the AI to send structured queries and receive responses in a machine-readable format.

Finally, the response processing system helps ensure that the retrieved data is formatted correctly and presented to the user in a meaningful way.

Step 1. Recognizing the need for a tool

Let's say a user asks an LLM "What's the weather in San Francisco right now?" The AI uses natural language understanding to recognize that real-time weather data is needed, which cannot be derived from its static knowledge base.

A unique tool call ID is assigned automatically to a request made by a model to use a tool, which acts as a tracking number to link the request with its corresponding result.

Step 2. Selecting the tool

The AI identifies the best tool for the task, in this case checking a current weather database. This step helps ensure that the retrieved information is accurate and relevant.

Each tool contains metadata and structured information such as a unique tool name (or function name), which helps the model and system identify it correctly. Other metadata include description, tool parameters and required input and output types.

The model performs a tool choice after determining that data must be obtained from a selection of available tools.

Templates are structured prompt formats that tell the model which tool to use and what arguments (or "args") to provide, allowing for more controlled and structured interactions with APIs.

In the context of tool calling, args refer to the structured inputs passed to a tool or function when it is started by a generative model. These arguments define the parameters that the tool requires to execute properly.

Combining tool calling with retrieval augmented generation (RAG) enhances AI capabilities by allowing systems to retrieve both structured and unstructured data before generating structured outputs.

This approach enhances contextual relevance by fetching the most pertinent data before generating a response, leading to more informed and accurate outputs.

It also minimizes API overhead by consolidating multiple retrievals into a single step, reducing latency and costs. RAG is more flexible than traditional tool calls, allowing models to pull from diverse sources and making it highly adaptable across different domains.

Unlike the rigid structure of traditional tool use, RAG enables more fluid integration of retrieved knowledge with reasoning and generation, resulting in more dynamic and insightful responses.

Step 3. Constructing and sending a query

The AI then formulates a structured request that the tool or API can understand.

Each tool is associated with specific tool functions, which define what the tool does. These functions rely on an API reference, which provides documentation on how to interact with the tool's API, including endpoint URLs, request methods and response formats.

To access an external API, many services require an API key, a unique identifier that grants permission to make requests. When the tool is selected and the parameters are set, an API call is made to fetch the requested data. This request is typically sent over HTTP to an external server.

Step 4. Receiving and processing the response

The external tool returns data. The AI must then parse the tool results. For a weather request, the API might respond with a JSON schema object containing temperature, humidity and wind speed. The AI filters and structures this data to summarize a meaningful response for the user.

Step 5. Presenting the information or taking action

The AI delivers the processed information in an intuitive manner. If the request involves automation, such as setting a reminder, the AI would confirm that an action has been scheduled.

Step 6. Refining the search

If the user requests more details or modifications, the AI can repeat the process with an adjusted query, helping to ensure that it continues to refine its response based on user needs.

LangChain is commonly used in tool calling by providing an open source framework for integrating external tools, APIs and functions with LLMs. It helps manage tool execution, input or output handling and context-aware decision-making.

For example, LangChain handles function arguments with a parser for user queries, extracting relevant parameters and formatting them correctly for the tool. Unlike simple tool calling, LangChain can store and recall previous tool outputs, enabling better multiturn interactions.

LangChain allows for the combination of multiple tools in a sequence, enabling more complex agentic workflows. For example, it can first retrieve data from the weather API and then use a separate tool to recommend clothing based on the forecast.

Types of tool calling

Tool calling allows LLMs to do all sorts of tasks. There are limitless use cases for AI applications that use tool calling, but here are 5 common categories with some real-world examples.

Information retrieval and search

AI fetches real-time data from the web, news sources, academic databases or financial markets. For example, an AI chat model can call a search API to provide the latest stock prices or AI research articles and deliver the information through a chatbot.

Code execution

This allows AI to perform complex calculations or run scripts using mathematical engines such as Wolfram Alpha or Python execution environments. This is useful for solving equations, running simulations or executing small code snippets.

Process automation

AI automates workflows such as scheduling meetings, sending emails or managing to-do lists through integrations with platforms such as Google Calendar and Zapier. AI agents can interact with CRM, finance and analytics tools such as Salesforce and QuickBooks, allowing businesses to automate processes including customer data retrieval or financial reporting.

Smart devices and IoT monitoring

Agentic AI systems can monitor and control home automation systems, industrial IoT devices and robotics. We can easily imagine that one day entire end-to-end workflows are handled by autonomous agents.