AI agent frameworks: Choosing the right foundation for your business

From a single artificial intelligence (AI) agent that monitors and flags fraudulent transactions for financial institutions to a multiagent system for supply chain management that tracks inventory levels and forecasts demand, agentic AI can be a boon for businesses. So how can enterprises get started with AI agents? This is where AI agent frameworks come in.

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.

AI agent frameworks: A foundational structure for agentic AI

AI agents are programs that can autonomously perform a task on behalf of a user. These AI systems first devise a plan with a series of steps to accomplish a complex task.

Then, they use function calling to connect to external tools—such as application programming interfaces (APIs), data sources, web searches and even other AI agents—that can help fill any gaps in their knowledge.

After executing their plan of action, autonomous agents learn from feedback and store learned information in memory to improve future performance.

Organizations can build AI agents from scratch by using programming languages such as Python or JavaScript. However, a quicker, more scalable approach involves using AI agent frameworks.

Agentic frameworks are the building blocks for developing, deploying and managing AI agents. These software platforms have built-in features and functions that help streamline and speed up the process, including:

- A predefined architecture that outlines the structure, characteristics and capabilities of agentic AI

- Communication protocols that facilitate the interaction between AI agents and human users or other agents
- Task management systems to coordinate tasks
- Integration tools for function calling
- Monitoring tools to track agentic AI performance

Factors to consider when choosing an AI agent framework

Before diving into the world of AI agents, think about your organization's goals and use cases. The ideal framework strikes a balance between your technical capabilities, your short-term requirements and your long-term objectives.

Here are a few aspects to factor in when selecting an AI agent framework:

- Complexity
- Data privacy and security
- Ease of use
- Seamless integration
- Performance and scalability

Complexity

Identify the tasks that you want an AI agent to fulfill and how complex these tasks are. Determine whether you need a simple implementation with only a single agent or a multiagent ecosystem.

For multiagent environments, map out the agent interactions required and where human intervention is still needed.

In the customer support realm, for instance, a single AI agent can help classify the severity of issues that come in. However, if you're aiming for a more robust workflow, consider creating a multiagent system with different agents to troubleshoot issues, suggest fixes and assign complicated cases to other AI or human agents.

Data privacy and security

Data privacy and security must be top of mind when selecting an agentic framework. Verify the security policies and measures of your framework of choice, including encryption for data at rest and in transit, access controls and removing any sensitive information.

Ease of use

Consider your development team's skill level. A beginner-friendly AI framework such as CrewAI, for example, has a no-code interface for rapid prototyping and ready-made AI agent templates for swift deployment.

More experienced AI developers might go for advanced agent frameworks such as LangGraph that offer low-level control and customizable code options.

Seamless integration

Evaluate agentic AI frameworks based on their compatibility with your existing tech stack. Check how well your choice of framework integrates with your current data sources, infrastructure and tools.

Figure out how agentic AI will be deployed to your environment—be it on-premises or in the cloud—and if a small-scale or large-scale deployment is required.

Performance and scalability

Appraise the performance of your chosen AI agent framework. Think about response time or latency for real-time applications, and assess if performance degrades when processing huge volumes of data or multiple concurrent requests. And while the focus might be on the short term, think about how the framework scales as your business grows.

Popular AI agent frameworks

Agentic AI is still in its early stages. As the technology behind AI agents evolves, so too will the frameworks underlying them. Here are some currently popular AI agent frameworks:

AutoGen

AutoGen is an open-source framework from Microsoft for creating multiagent AI applications to perform complex tasks. Its architecture consists of 3 layers:

- Core is a programming framework for developing a scalable and distributed network of agents, with tools for tracing and debugging agent workflows. It employs asynchronous messaging, supporting both request-response and event-driven agent interactions.
- AgentChat is built on top of Core and can be used to craft conversational AI assistants. It’s the proposed starting point for beginners, offering default single agents and multiagent teams with predefined behaviors and interaction patterns.
- Extensions is a package containing implementations of Core and AgentChat components to further expand their capabilities and interface with external libraries and other services. You can use built-in extensions and those developed by the AutoGen community, or even create your own.

AutoGen also provides 2 handy developer tools: AutoGen Bench for assessing and benchmarking agentic AI performance and AutoGen Studio for a no-code interface to develop agents. AutoGen is available to access on GitHub.

CrewAI

CrewAI is an orchestration framework for multiagent AI solutions. Like AutoGen, CrewAI is open source.

CrewAI’s role-based architecture treats agentic AI as a “crew” of “workers.” Here are the core components of a crew:

- Agents are assigned specialized roles while still collaborating on complex workflows. Developers can use natural language to outline an agent's role, goal and backstory.
- Tasks define the specific responsibilities of each agent. Developers can also use natural language to describe the task and expected output for each agent.
- A process identifies how agents work together and how tasks are executed. It can either be sequential, with tasks completed according to a preset order or hierarchical, with a custom manager agent overseeing task delegation, execution and completion.

One of CrewAI's collection of examples includes a stock market analysis crew. This crew collaborates in a sequential manner, with a market analyst agent tasked to analyze data for a particular stock, a researcher agent tasked to gather supporting information that validates the data analysis, and a strategy agent tasked to create a step-by-step action plan based on the analysis and supporting data.

CrewAI supports connections to various large language models (LLMs), including Anthropic's Claude, Google's Gemini, Mistral's AI models, OpenAI's GPT models and the foundation models in IBM® watsonx.ai™.

The framework also has a suite of retrieval augmented generation (RAG) tools to search different data sources.

CrewAI is available to access on GitHub.

LangChain

LangChain is another open-source framework for building LLM-powered applications, including chatbots such as ChatGPT and AI agents.

It employs a modular architecture, with each module representing abstractions that encapsulate the complex concepts and steps necessary to work with LLMs.

These modular components can then be chained together to create AI applications.

LangChain is useful for developing simple AI agents with straightforward workflows. It provides support for vector databases and utilities for incorporating memory into applications, as a result retaining history and context.

Its LangSmith platform allows for debugging, testing and performance monitoring.

LangChain is available to access on GitHub.

LangChain4j

LangChain4j is an open-source Java library that makes it simpler and easier to incorporate large language models into Java applications. A unified API provides access to common LLMs and vector databases.

It features capabilities ranging from low-level prompt templating, chat memory management and function calling to high-level patterns like agents, retrieval augmented generation (RAG) and tools (including support for the Model Context Protocol). It also facilitates integrations with various enterprise Java frameworks.

LangChain4j follows a modular design:

- Thelangchain4j module contains high-level features such as the unified API, chat memory implementations and tools like document loaders.
- The langchain4j-core module outlines core abstractions and their corresponding APIs.
- A range oflangchain4j-{integration} modules offer integration with different LLMs and vector databases.

Meanwhile, thelangchain4j-agentic module enables building agentic AI applications and orchestrating multi-agent workflows. It allows for context and memory engineering and managing tool calling. Additionally, thelangchain4j-agentic-a2a module supports the use of the Agent2Agent protocol to connect multiple agents together in a distributed agentic system.

LangChain4j is available to access on GitHub.

LangGraph

LangGraph lives within the LangChain ecosystem. The framework excels at orchestrating complex workflows for multiagent systems.

It applies a graph architecture, wherein the specific tasks or actions of AI agents are depicted as nodes, while the transitions between those actions are represented as edges.

A state component maintains the task list across all interactions. This type of architecture is suitable for cyclical, conditional or nonlinear workflows.

For example, an airline might want to build a travel assistant AI agent that helps users find and book flights. Using LangGraph, each of those actions will be represented as nodes, and those nodes can have multiple agents performing particular tasks.

A human-in-the-loop step can be added so users can choose a flight from the search list, and if nothing suits their preferences, the travel assistant agent can easily change back to the “find flights” node and redo the search.

LangGraph is available to access on GitHub.

LlamaIndex

LlamaIndex is an open-source data orchestration framework for building generative AI (gen AI) and agentic AI solutions. It offers prepackaged agents and tools and recently introduced workflows, a mechanism for developing multiagent systems.

Here are the main elements that make up a workflow in LlamaIndex:

- Steps are the specific actions of an agent. These are the basic components of a workflow.
- Events trigger steps and are the means by which steps communicate.
- Context is shared across the workflow so steps can store, retrieve and pass data and maintain state throughout their run.

This event-driven architecture enables workflow steps to be accomplished asynchronously. This means that, unlike a graph architecture, the paths between steps don’t need to be defined, resulting in more flexible transitions between agent actions.

As such, LlamaIndex workflows are well-suited for more dynamic AI agent applications that need to loop back often to previous steps or branch to several steps.

LlamaIndex is available to access on GitHub.

Semantic Kernel

Semantic Kernel is an open-source development kit from Microsoft for building enterprise-grade generative AI applications. Its Agent Framework, currently marked as experimental, provides core abstractions for creating agents.

It has 2 built-in agent implementations: a chat completion agent and a more advanced assistant agent.

Multiple agents can be orchestrated through group chats or by using Semantic Kernel's Process Framework (also marked as experimental) for more complex workflows.

A process consists of steps, which represent the tasks assigned to AI agents, and outline how data flows between steps.

Semantic Kernel is available to access on GitHub.

For more informed decision-making, consider experimenting with your preferred frameworks. Start small with a simple, single-agent implementation to test how each framework operates and how it compares to others.

The right agentic framework aligns with your enterprise needs and can help craft AI agents that automate workflows, leading to more efficient business processes.