What is crewAI?

Authors

Vanna Winland
AI Advocate & Technology Writer

Meredith Syed
Technical Content, Editorial Lead
IBM

Anna Gutowska
AI Engineer, Developer Advocate

crewAI is an open source multiagent orchestration framework created by João Moura. This Python-based framework leverages artificial intelligence (AI) collaboration by orchestrating role-playing autonomous AI agents that work together as a cohesive assembly or "crew" to complete tasks. The goal of crewAI is to provide a strong framework to automate multiagent workflows.[1]

The term "crew" refers to AI agents that work together to autonomously delegate tasks and ask questions among themselves, similar to a real-life work crew. Each multiagent crew is composed of complementary role-playing AI agents who leverage existing and custom tools to complete an assigned set of tasks. Language models act as a reasoning engine for agents by selecting a series of actions.[2] crewAI's agents can be configured to use any open source large language model (LLM) or application programming interface (API).

Recent research expands the scope of LLM models beyond just text generation, demonstrating that they can serve as versatile agents for conversational interactions, decision-making and task completion.[3] In the burgeoning field of AI and research surrounding AI agents and agentic frameworks, multiagent frameworks including crewAI have emerged onto the gen AI scene.

The next generation of AI applications will use agentic architecture to build autonomous agent-based systems.[4] These agentic frameworks tackle complex tasks for a multitude of AI solutions by enhancing generative AI tasks. For example, AI chatbots can be a modality for implementing agentic

AI frameworks. Agentic chatbots, unlike nonagentic ones, can use their available tools, plan actions before running them and hold memory. These capabilities produce more refined and meaningful conversations.

The latest AI News + Insights

Discover expertly curated insights and news on AI, cloud and more in the weekly Think Newsletter.

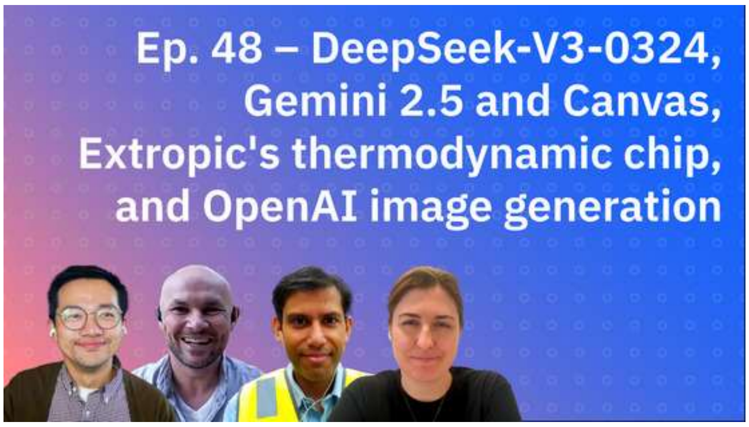
Subscribe today

AI agent (agentic) frameworks

Agentic frameworks are AI agent architectures that use tool calling and orchestration for AI applications. Agentic systems use planning, iterative refinement, reflection and other control mechanisms to fully leverage the model's built-in reasoning capabilities to complete tasks end-to-end.[5] Implementing AI agents within AI systems automate the necessary processes for gen AI applications to function.

Agentic frameworks also provide enhanced learning and performance abilities. Fine-tuning LLMs for custom decision-making tasks is both resource intensive and can diminish the models' generalization capabilities.[6] AI agents can learn from their previous actions and experiences, lightening the computational expense needed for fine-tuning models.

Mixture of Experts | 28 March, episode 48


Ep. 48 – DeepSeek-V3-0324,
Gemini 2.5 and Canvas,
Extropic's thermodynamic chip,
and OpenAI image generation

Decoding AI: Weekly News Roundup

Join our world-class panel of engineers, researchers, product leaders and more as they cut through the AI noise to bring you the latest in AI news and insights.

Watch the latest podcast episodes →

Agentic system architecture

Agentic frameworks can use single-agent or multiagent systems.

Single versus multiagent

Single-agent frameworks rely on one language model to run a diverse range of tasks and responsibilities. The agent is supplied with a system prompt and the necessary tools to complete their tasks such as search, APIs and even other agents. Although single-agent systems can interact with other agents through tooling, they do not cooperate in the same way that multiagent systems do.

In single-agent systems there is no feedback mechanism from other AI agents; therefore, options for human feedback to guide the agent are recommended to improve accuracy overtime. Single-agent architectures perform best for well-defined problems where feedback from other agents or users is unnecessary.[7]

Rather than trying to encompass all capabilities within a single model, multiagent systems (MAS) divvy up tasks among several specialized agents. Multi-agent architectures involve two or more agents, which might use the same language model or different ones. No matter the size, the agents work within the same environment to model each other's goals, memory and plan of action. These architectures demonstrate notable advantages over chain-of-thought (CoT) prompting, where the model needs to break down tasks into a series of steps.[8] Multiagent architectures tend to thrive more when collaboration and multiple distinct execution paths are required.

The best agent architectures to use depend on the specifics of the overall application and use case. Single-agent systems are best at solving narrow problems. One can think of agents as problem-solvers. Some problems require the individual capabilities of one specialized agent, others might require a team of problem-solvers, or a team of multiple agents. Multiagent systems are a team of agents that work together to solve problems that are beyond the individual capabilities or knowledge of each agent. Multi-agent systems can solve problems that are too large for single-agent systems. Research suggests that multiagent systems have significant advantages including enhanced speed and reliability and tolerate uncertain data and knowledge.[9] Key benefits of multiagent systems similar to crewAI include agent collaboration, autonomous workflows and scalability.

Agent collaboration

AI agents can be optimized via their versatile customizable parameters. Each agent is assigned a persona that outlines their role along with any specific instructions for their behavior.[10] Multiagent frameworks use agents' abilities to manage tasks within a team while acting in their specific roles and interacting with each other. The compositions of these teams can be adapted and optimized depending on the application and overall goals.

One way this is being implemented is via collaborative generative agents. Multiagent frameworks are able to provide the essential faculties needed for effective collaboration.[11] Some multiagent frameworks provide templates for agent collaboration based on the overall goal. crewAI facilitates agent collaboration by allowing users to assemble agents into teams, or crews that work to execute a common goal or task.

Autonomous behavior

Autonomous AI agents can complete a task or series of complex tasks undirected. The potential of LLM-based autonomous agents is recognized as a leading approach toward achieving artificial general intelligence (AGI).[12] These LLM-based agents can execute tasks via self-directed planning and actions. While they have demonstrated impressive capabilities, AI agents still face challenges in expanding their competencies for tasks that require more complex forms of reasoning.[13] Agentic systems help mitigate these challenges by providing a framework for autonomous workflows. crewAI offers autonomous behavior through its hierarchical process that uses an autonomously generated manager agent that oversees the execution and allocation of agent tasks.

Scalability

Multiagent systems must scale in several different dimensions. Those dimensions include when the total number of agents increases across a system or application, the diversity of the agents increases, and the size of the data the agents are operating on or with increases.[14] Many multiagent frameworks come with tooling such as monitoring and metrics to help assess whether the system is scaling successfully. crewAI allows integration with third-party resource monitoring and metric tools to set up observability and evaluations for LLMs, LLM frameworks and vector databases.

AI agents

AI agents are LLM-based systems or programs that can be developed to perform various complex tasks. Agents possess memory and planning capabilities that allow them to make independent decisions and take actions based on their previous experience.[15] Agents enhance the traditional LLMs capabilities by using LLM outputs to invoke other software tools (such as data retrieval) and loop the outcomes back into an LLM until the overarching objective is fulfilled. What sets AI agents apart from traditional LLMs is their ability to navigate, interact and adapt to their environment through action planning, memory utilization and tool cooling. Agentic systems provide tooling and orchestration for agents to run machine learning algorithms associated with their tasks.

How crewAI works

crewAI is built on top of LangChain with a modular design principle in mind. Its main components consist of agents, tools, tasks, processes and crews.

Agents

Agents are the fundamental components of the crewAI framework. Each agent is an autonomous unit with different roles that contribute to the overall goal of the crew. Each agent is programmed to perform tasks, handle decision-making and communicate with other agents.

crewAI encourages users to think of agents as members on a team. Agents can have different roles such as ‘Data Scientist’, ‘Researcher’ or ‘Product Manager’. The multiagent team effectively collaborates to perform automated workflows.

This form of multiagent system aims to enhance the LLMs’ reasoning abilities through interagent discussions by utilizing a role-playing structure to facilitate complex problem-solving.[16] Agents engage with one another through crewAI's inherent delegation and communication mechanisms giving them the innate ability to reach out to one another to delegate work or ask questions.

Agent attributes

The agent’s goals and characteristics are defined by attributes. crewAI’s agents have three main attributes **role**, **goal** and **backstory.**

For example, an instantiation of an agent in crewAI may look like this:

```
  agent = Agent(
      role= 'Customer Support',
      goal= 'Handles customer inqueries and problems',
      backstory= 'You are a customer support specialist for a chain restaurant. You are responsible
for handling customer calls and providing
      customer support and inputting feedback data.'
      )
```

crewAI offers several optional parameters including attributes to choose what LLM and tooling dependencies the agent uses.[17]

Tools

Tools are skills or functions that agents use to perform different tasks. Users can leverage both custom and existing tools from the crewAI Toolkit and LangChain Tools.

Tools extend the capabilities of agents by enabling them to perform a broad spectrum of tasks including error handling, caching mechanisms and customization via flexible tool arguments.

crewAI tools

All tools contain error handling and support caching mechanisms.

The crewAI Toolkit contains a suite of search tools that use the Retrieval-Augmented Generation (RAG) methodology within different sources. A few examples include:

JSONSearchTool: Perform precision searches within JSON files.
GithubSearchTool: Search within GitHub repositories.
YouTubeChannelSearchTool: Search within YouTube channels.

Beyond RAG tools, the kit also contains various web-scraping tools for data collection and extraction.

LangChain tools

crewAI offers simple integration with LangChain tools. Here are a few examples of available built-in tools from LangChain:

Shell (bash): Gives access to the shell, enabling the LLM to execute shell commands.[18]
Document comparison: Use an agent to compare two documents.[19]
Python: Enable agents to write and execute Python code to answer questions.[20]

Custom tools

Users can create their own tools to further optimize agent capabilities. As part of the crewAI tools package, users can create a tool by defining a clear description for what the tool will be used for. The agent will use the user-defined description to use the custom tool. Custom tools can optionally implement a caching mechanism that can be fine-tuned for granular control.

Tasks

Tasks are specific assignments completed by agents. Details for execution are facilitated through task attributes. Multiple agents can be assigned to work together to complete the same task.

Task attributes

The required task attributes include **description, agent** and **expected output**. These attributes define the scope of the task, the responsible agent and the goal. A task can either be directly assigned to an agent or handled through crewAI’s hierarchical process that decides based on roles and availability.

Here is an example of a task:

```
data_collection = Task(
    description= ‘Gather data from customer interactions, transaction history, and support tickets’
    expected_output= ‘An organized collection of data that can be preprocessed’,
    agent=data_science_agent,
)
```

Optional task attributes include tool integration, asynchronous execution for concurrency and output formats including JSON, Pydantic models and file outputs for task results.

Task features

Task features include tool integration, asynchronous execution, human input review and output customization.

The results of a task can establish context for a future task. For example, the outputs from a “research” task can be used as context to complete a “writer” task. Consider a simple example, a team of two agents, one “research agent” and one “writer” agent. The research agent is tasked with finding examples of top generative AI use cases, the writer agent can use the resulting research as context to perform the task of writing a short blog about the same or similar topic.

Tasks can be defined to execute asynchronously. This is useful for tasks that take a long time to be completed or aren’t necessary for the next tasks to be performed. The context attribute can be used to define in a future task that it should wait for the output of the asynchronous task to be completed.[21]

Processes

Processes enable individual AI agents to operate as a cohesive unit by orchestrating the execution of tasks. Processes in agentic frameworks define how agents will work together and what tasks they will be assigned. crewAI compares processes to project management because they ensure that tasks are distributed and executed efficiently and remain aligned with a predefined strategy to complete the goal.

crewAI includes two process implementations: sequential and hierarchical and plan for a third called the consensual process. Processes can be assigned to a crew of agents to enable them to operate as a cohesive unit. When assigning a process to a crew, the process type sets the execution strategy.

> **Sequential:** The sequential process is similar to a dynamic team workflow. Tasks are executed according to the predefined order in the task list, with the output of one task serving as the context for the next.
> **Hierarchical:** The hierarchical process emulates a corporate hierarchy. crewAI autonomously generates a manager for users by leveraging a manager language model tailored for the manager agent.[22] The manager agent oversees task execution and allocates tasks to agents based on their capabilities, reviews outputs and assesses task completion. This process is an example of AI agents working autonomously and collaboratively to complete a set of tasks.
> **Consensual (planned):** At the time of writing, the consensual process is not currently implemented in the codebase but aims to provide a way for collaborative decision-making among agents on tasks execution. This process introduces a democratic approach to task management.

Crews

A crew embodies a collective ensemble of agents collaborating to accomplish a predefined set of tasks.[23] Each crew defines the strategy for task execution, agent execution and the overall workflow. Crews have several attributes that help assemble agents with complementary roles and tools, assign tasks and select a process that dictates their execution order and interaction.[24]

Crew attributes

Users choose and define a list of agents to work together as a crew. Crews are assigned a list of tasks to complete. Optional attributes define the strategy for execution, agent collaboration and the overall workflow.

Here is an example of a crew consisting of two agents with the goal of working together to collect and organize customer support data:

```
my_crew = Crew(
    agents=[data_science_agent, customer_support_agent],
    tasks=[customer_support_task, data_collection_task],
    process=Process.sequential,
    full_output=True,
    verbose=True,
)
```

Additional attributes include callback functions, language and memory settings and options to set a manager agent and LLM to be used depending on the process flow (for example, sequential, hierarchical). Once a crew is assembled, the workflow is initiated via a start-up method. crewAI provides several start-up methods to control the process including asynchronous and individual task execution.[25]

Connect to any LLM

crewAI can connect to any LLM through a variety of connection options. By default, agents will use OpenAI's GPT-4 model for language processing, however, crewAI offers flexibility in connecting to various LLMs, including models such as the IBM Granite™ series. Local models can be connected through ollama or other open APIs. Examples of API key configurations and tutorials on connecting to several LLMs are provided in the crewAI docs. crewAI is compatible with all LangChain LLM components that give all LLMs basic support for a runnable interface.

crewAI use cases

AI agent frameworks such as crewAI serve as foundational tools for researchers and developers to create intelligent systems across various domains, ranging from agentic AI chatbots to complex multiagent systems.

Several real-world examples include projects such as building interactive landing pages and using a crew to automate the process of boosting social media presence. A collection of several real-world examples exist in a GitHub repository organized by Moura titled "crewAI-examples" for users to test out for themselves.[26] These examples also include introductions for beginners to use the framework.

Here's a list of a few of those examples and other use cases emerging from the crewAI community:

- **Content planning and creation:** One use case uses crewAI and groq, a natural language model, to create a team of specialized agents to create engaging and factually accurate content on a given topic.[27]
- **Automate checking and drafting emails:** Designed as an introduction for beginners, a crew of agents complete the tasks of analyzing and filtering emails, pulling the full threads, researching and creating email drafts using the LangGraph library to automate the multiagent workflows.[28]
- **Stock analysis:** Agents are given specific roles to collaborate to give a complete stock analysis and investment recommendation using GPT 3.5 instead of the default GPT-4.[29]

Other multiagent frameworks

crewAI compares itself to multiagent frameworks including AutoGen and ChatDev. The biggest advantage of crewAI is the combination of what these two frameworks do well individually. crewAI combines the flexibility of AutoGen's conversational agents with the structured processes approach of ChatDev.[30]

crewAI versus AutoGen

AutoGen is Microsoft's open source agentic framework that uses natural language processing (NLP) algorithms for conversational AI agents. While both platforms are used in similar applications, they each have their respective pros and cons. Both are flexible systems with customizable AI agents capable of collaboration. crewAI provides a simpler way to orchestrate agent interactions by providing customizable attributes that control the application's processes. Autogen requires more programming setup to achieve this. AutoGen offers a built-in way to quickly execute LLM-generated code.[31] crewAI does not currently offer tooling for this ability, but it is possible to do with additional programming setup.

crewAI versus ChatDev

ChatDev is an open source platform that uses role-playing multiagent collaboration, including crewAI. ChatDev's process structure is rigid, therefore limiting customization and hindering scalability and flexibility for production environments. Frameworks like crewAI are designed to integrate with third-party applications and customizable workflows for dynamic and adaptable environments. A unique functionality of ChatDev is that it extends itself as a browser extension to chain together conversations across various agents within a web browser.[32]

As a multiagent orchestration framework, crewAI provides another innovation towards the goal of artificial intelligence. Agentic architectures will enhance the performance and capabilities of AI agents, enabling LLM applications to carry out tasks beyond language generation.