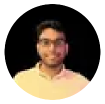
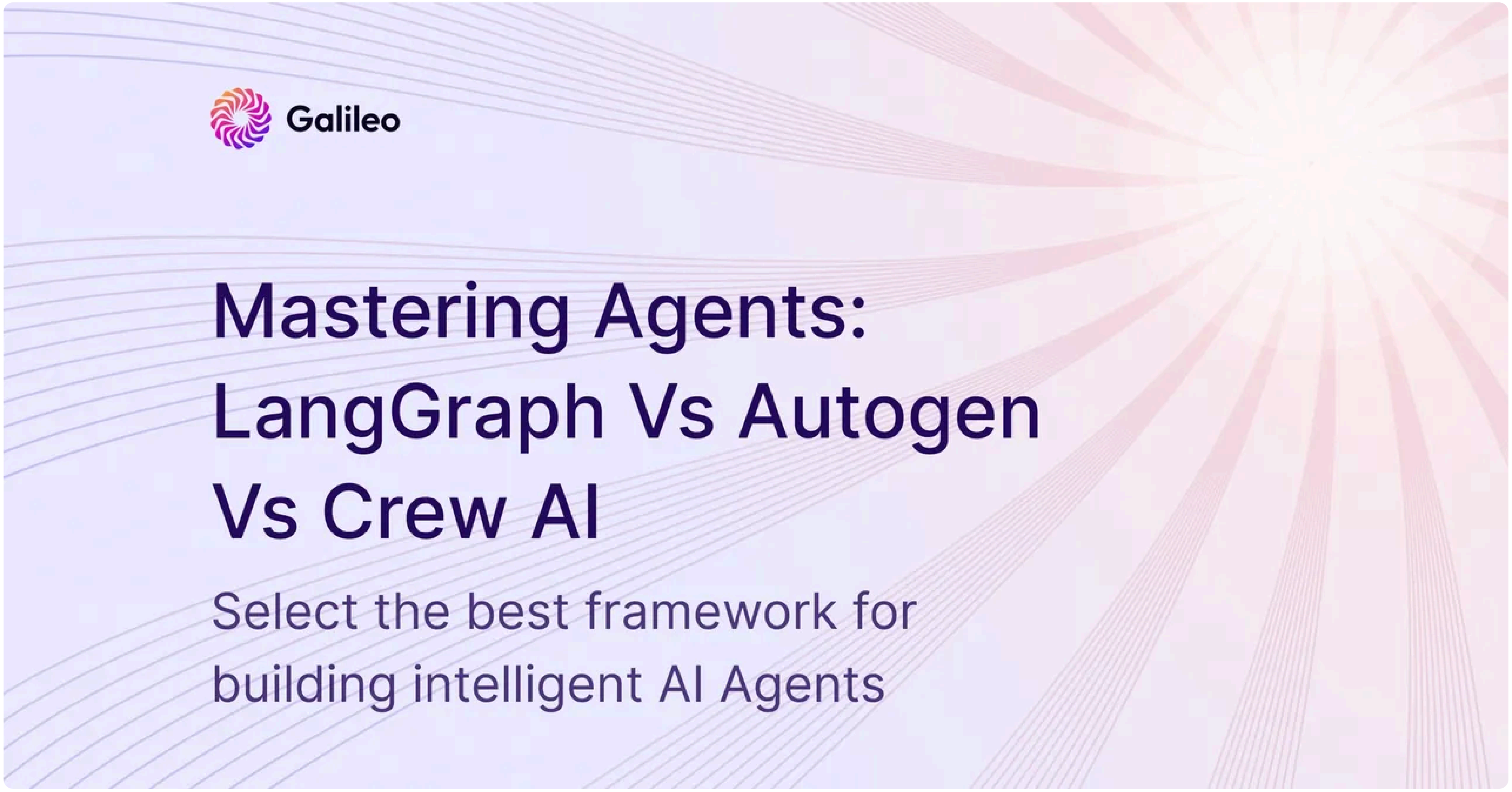
# Mastering Agents: LangGraph Vs Autogen Vs Crew AI


Pratik Bhavsar
Galileo Labs


Galileo
Mastering Agents: LangGraph Vs Autogen Vs Crew AI
Select the best framework for building intelligent AI Agents

10 min read

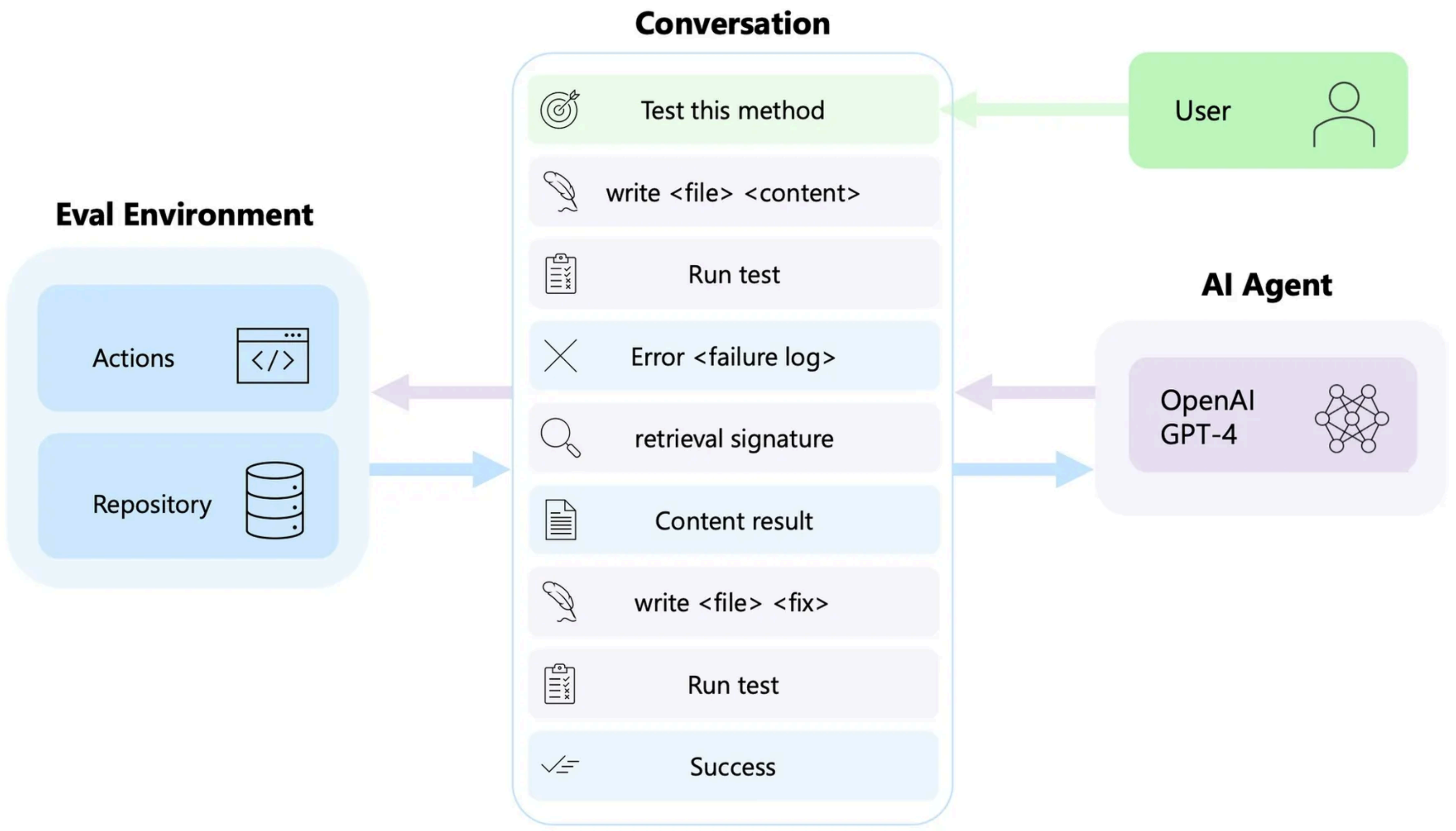
September 05 2024

AI agents are on the rise, playing a crucial role in automating processes that were once thought impossible. These agents leverage LLMs to perform a wide range of tasks, from generating sales leads to making investment decisions. However, the choice of framework for building these agents can significantly affect their efficiency and effectiveness. In this blog, we will evaluate three prominent frameworks for building AI agents — LangGraph, Autogen, and Crew AI — to help you make an informed choice.

## What is an Agent?

An agent in AI is a software application that uses LLMs to perform specific tasks *autonomously*. These tasks can range from answering research questions to invoking backend services. Agents can be particularly useful in scenarios requiring open-ended answers, where they can provide surprisingly effective solutions.

Customer support agents are capable of addressing customer inquiries, supplying information, and autonomously resolving issues. While code generation agents can generate, debug, and run code snippets, assisting developers in automating repetitive tasks.



AutoDev: Automated AI-Driven Development

## When to Use Agents

Agents are highly beneficial when tasks require complex decision-making, autonomy, and adaptability. They excel in environments where the workflow is dynamic and involves multiple steps or interactions that can benefit from automation. For instance, in customer support, agents can handle a wide range of queries, provide real-time assistance, and escalate issues to human operators when necessary. This not only improves efficiency but also enhances the customer experience by providing timely and accurate responses.

In research and data analysis, agents can autonomously gather, process, and analyze large volumes of data, providing valuable insights without human intervention. They are also useful in scenarios requiring real-time data processing, such as financial trading, where agents can make split-second decisions based on market conditions.

Moreover, agents are beneficial in educational applications, where they can provide personalized learning experiences, adapt to the student's pace, and offer instant feedback. In software development, agents can assist in code generation, debugging, and testing, significantly reducing development time and improving code quality. The ability of agents to learn from interactions and improve over time makes them invaluable in environments where continuous improvement and adaptation are crucial.

## When Not to Use Agents

Agents offer numerous advantages but there are scenarios where their use may not be the right choice.

To be clear, in scenarios where tasks are straightforward, infrequent, or require minimal automation, the complexity of implementing agents may not be justified. Simple tasks that existing software solutions can easily manage do not necessarily benefit from the added complexity of agent-based systems. In such cases, traditional methods are more efficient and cost-effective.

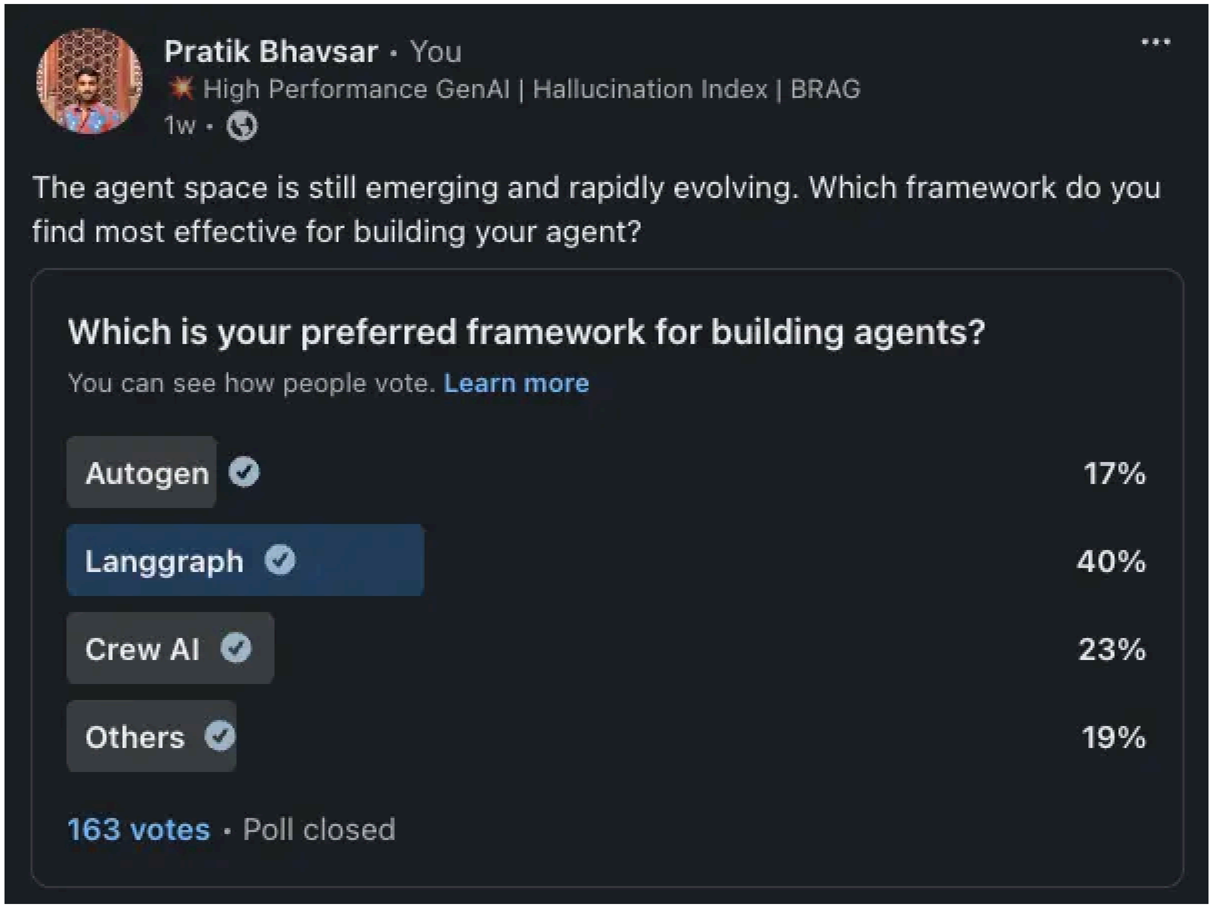
Additionally, tasks that require deep domain-specific knowledge and expertise, which cannot be easily encoded into an agent, may not benefit from automation. For instance, complex legal analysis, intricate medical diagnoses, or high-stakes decision-making in uncertain environments often require the expertise and intuition of seasoned professionals. In such cases, relying solely on agents can lead to suboptimal or even harmful outcomes.

Agents are also not well-suited for tasks that require a high level of human empathy, creativity, or subjective judgment. For example, in fields such as psychotherapy, counseling, or creative writing, the nuances of human emotions and creativity are difficult for agents to replicate. In these cases, human interaction is irreplaceable and essential for achieving the desired outcomes.

Implementing agents requires a significant investment in terms of time, resources, and expertise. For small businesses or projects with limited budgets, the cost of developing and maintaining agents may outweigh the benefits. Furthermore, in highly regulated industries, the use of agents may be restricted due to compliance and security concerns. Ensuring that agents adhere to stringent regulatory requirements can be challenging and resource-intensive.

## LangGraph vs Autogen vs Crew AI

Having clarified when to utilize an agent, let’s shift our focus to the primary topic: the most widely adopted frameworks in the industry. We conducted a poll and identified three frameworks that are most commonly used. Let’s begin with a high-level overview of these frameworks.


Pratik Bhavsar • You
High Performance GenAI | Hallucination Index | BRAG
1w •

The agent space is still emerging and rapidly evolving. Which framework do you find most effective for building your agent?

Which is your preferred framework for building agents?
You can see how people vote. Learn more

| | |
|---|---|
| Autogen ✓ | 17% |
| Langgraph ✓ | 40% |
| Crew AI ✓ | 23% |
| Others ✓ | 19% |

163 votes • Poll closed

### LangGraph

LangGraph is an open-source framework designed by Langchain to build stateful, multi-actor applications using LLMs. Inspired by the long history of representing data processing pipelines as directed acyclic graphs (DAGs), LangGraph treats workflows as graphs where each node represents a specific task or function. This graph-based approach allows for fine-grained control over the flow and state of applications, making it particularly suitable for complex workflows that require advanced memory features, error recovery, and human-in-the-loop interactions. LangGraph integrates seamlessly with LangChain, providing access to a wide range of tools and models, and supports various multi-agent interaction patterns.

### Autogen

Autogen is a versatile framework developed by Microsoft for building conversational agents. It treats workflows as conversations between agents, making it intuitive for users who prefer interactive ChatGPT-like interfaces. Autogen supports various tools, including code executors and function callers, allowing agents to perform complex tasks autonomously. The framework is highly customizable, enabling users to extend agents with additional components and define custom workflows. Autogen is designed to be modular and easy to maintain, making it suitable for both simple and complex multi-agent scenarios.

### Crew AI

Crew AI is a framework designed to facilitate the collaboration of role-based AI agents. Each agent in Crew AI is assigned specific roles and goals, allowing them to operate as a cohesive unit. This framework is ideal for building sophisticated multi-agent systems such as multi-agent research teams. Crew AI supports flexible task management, autonomous inter-agent delegation, and customizable tools.

Let's compare LangGraph, Autogen, and Crew AI across several key aspects for practical consideration.

## Ease of Usage

Ease of usage determines how quickly and efficiently a developer can get started with a framework. It affects the learning curve and the time required to build and deploy agents.

LangGraph: LangGraph treats workflows as graphs, which can be intuitive for those familiar with data processing pipelines. It uses directed acyclic graphs (DAGs) to represent workflows, making it easier to visualize and manage complex processes. However, it may require a deeper understanding of graph-based structures.

Autogen: Autogen treats workflows as conversations between agents. This conversational approach can be more intuitive for users who prefer chat interfaces. The framework handles the heavy lifting of managing agent interactions, making it easier to get started. It abstracts much of the complexity, allowing users to focus on defining tasks and interactions.

Crew AI: Crew AI focuses on role-based agent design, where each agent has specific roles and goals. This framework is designed to enable AI agents to operate as a cohesive unit, which can be beneficial for building complex, multi-agent systems. It provides a structured approach to defining and managing agents. It is very straightforward to get started with Crew AI.

Winner: Autogen and Crew AI have an edge due to their conversational approach and simplicity.

## Tool Coverage

Tool coverage refers to the range of tools and functionalities that the framework supports, which can enhance the capabilities of agents.

LangGraph: LangGraph integrates seamlessly with LangChain, providing access to a wide range of tools and models. It supports tool calling, memory, and human-in-the-loop interactions. This integration allows users to leverage a broad ecosystem of tools to extend the functionality of their agents.

Autogen: Autogen supports various tools, including code executors and function callers. The framework's modular design makes it easy to add and integrate new tools as needed.

Crew AI: Crew AI is built over Langchain giving access to all of their tools. Users can also define and integrate tools tailored to their specific needs.

> Winner: LangGraph and Crew have an edge due to their seamless integration with LangChain, which offers a comprehensive range of tools. All the frameworks allow the addition of custom tools.

## Memory Support

Memory support is crucial for agents to maintain context across interactions, enabling them to provide more coherent and relevant responses. There are different types of memory that agents can use:

Short-Term Memory: Temporarily stores recent interactions and outcomes, allowing agents to recall information relevant to the current context.

Long-Term Memory: Preserves valuable insights and learnings from past interactions, enabling agents to build and refine their knowledge over time.

Entity Memory: Captures and organizes information about specific entities (people, places, concepts) encountered during tasks, facilitating deeper understanding and relationship mapping.

Contextual Memory: Combines short-term, long-term, and entity memory to maintain the overall context of interactions.

LangGraph: LangGraph provides built-in short-term, long-term, and entity memory, enabling agents to maintain context across interactions. It supports advanced memory features like error recovery and time travel, allowing users to revisit and analyze previous states.

Autogen: Autogen supports memory through its conversation-driven approach. Agents can remember previous interactions, making them suitable for tasks requiring contextual awareness. The framework's design ensures that agents can maintain a coherent context throughout their interactions.

Crew AI: Crew AI offers a comprehensive memory system, including short-term, long-term, and entity memory. This allows agents to accumulate experiences and improve their decision-making over time. The memory system ensures that agents can maintain context and recall important information across multiple interactions.

> Winner: Both LangGraph and Crew AI have an edge due to their comprehensive memory system, which includes short-term, long-term, and entity memory.

## Structured Output

Structured output is important for ensuring that the responses generated by agents are well-organized and easily interpretable. Structured output can include JSON, XML, or other formats that facilitate further processing and analysis.

LangGraph: LangGraph allows nodes to return structured output, which can be used to route to the next step or update the state. This makes it easier to manage complex workflows and ensures that the output is well-organized.

Autogen: Autogen supports structured output through its function-calling capabilities. Agents can generate structured responses based on the tools and functions they use. This ensures that the output is well-defined and can be easily processed by other components.

Crew AI: Crew AI supports structured output by allowing agents to parse outputs as Pydantic models or JSON. This ensures that the output is well-organized and easily interpretable. Users can define the structure of the output to meet their specific requirements.

Winner: LangGraph and Crew AI has an edge due to its ability to define structured output.

## Documentation

Documentation quality affects how easily developers can understand and use the framework. Good documentation can reduce the learning curve and improve the overall developer experience.

LangGraph: LangGraph provides comprehensive documentation, including detailed guides and examples. The documentation is well-structured, making it easy for users to find the information they need. It covers various aspects of the framework, from basic concepts to advanced features.

Autogen: Autogen has documentation with numerous examples and tutorials. The documentation covers various aspects of the framework, making it accessible to both beginners and advanced users. It includes detailed explanations of key concepts and features.

Crew AI: Crew AI provides detailed documentation, including how-to guides and examples. The documentation is designed to help users get started quickly and understand the framework's core concepts. It includes practical examples and step-by-step instructions.

Winner: All of the frameworks have great documentation but it's easy to find more examples of LangGraph and Crew AI.

## Multi-Agent Support

Multi-agent support determines how well the framework can handle various interaction patterns between multiple agents. This includes hierarchical, sequential, and dynamic interaction patterns.

Grouping tools and responsibilities can yield better results, as an agent is more likely to succeed on a focused task than if it has to select from dozens of tools. Separate prompts can also enhance performance, allowing each prompt to have its own instructions and few-shot examples. Each agent can even be powered by a separate fine-tuned LLM, providing a helpful conceptual model for development. This approach allows for evaluating and improving each agent individually without disrupting the larger application.

LangGraph: LangGraph supports various multi-agent patterns, including hierarchical and dynamic group chats. It allows users to define complex interaction patterns between agents. The framework's graph-based approach makes it easy to visualize and manage these interactions. LangGraph prefers an approach where you explicitly define different agents and transition probabilities, representing them as nodes in a graph. This graphical approach provides the highest flexibility for constructing complex and opinionated workflows, where controlling the transition probabilities between nodes is crucial.

Autogen: Autogen emerged as one of the first multi-agent frameworks, framing workflows more as "conversations" between agents. This conversational approach provides flexibility in defining how agents interact with each other, supporting multiple conversation patterns, including sequential and nested chats. Autogen's design makes it easy to manage complex multi-agent interactions, allowing agents to collaborate effectively..

Crew AI: Crew AI supports role-based interactions and autonomous delegation between agents. It provides processes like sequential and hierarchical task execution to manage multi-agent interactions effectively. This ensures that agents can work together efficiently to achieve common goals. Crew AI is a higher-level framework compared to LangGraph, focusing on creating multi-agent "teams" that operate cohesively.

> Winner: LangGraph has an edge due to its graph-based approach, which makes it easier to visualize and manage complex interactions.

## Caching

Caching is useful in reducing latency and resource consumption of agents by storing and reusing previously computed results.

LangGraph: LangGraph supports caching through its built-in persistence layer. This allows users to save and resume graph execution at any point. The caching mechanism ensures that previously computed results can be reused, improving performance.

Autogen: AutoGen supports caching API requests so that they can be reused when the same request is issued.

Crew AI: All tools in Crew AI support caching, enabling agents to efficiently reuse previously obtained results, reducing the load on external resources and speeding up the execution time. You can also define finer control over the caching mechanism using the cache_function attribute of the tool.

> Winner: All of the frameworks support caching but LangGraph and CrewAI might have an edge.

## Replay

Replay functionality allows users to revisit and analyze previous interactions, which is useful for debugging and improving agent performance. It enables users to understand the decision-making process and identify areas for improvement.

LangGraph: LangGraph supports replay through its time travel feature. Users can rewind and explore alternative paths, making it easier to debug and experiment with different scenarios. This feature provides a detailed history of interactions, allowing for thorough analysis.

Autogen: Autogen does not have an explicit replay feature, but users can manually update the state to control the agent's trajectory. This allows for some level of replay functionality, although it may require more manual intervention.

Crew AI: CrewAI provides the ability to replay from a task specified from the latest crew kickoff. Currently, only the latest kickoff is supported and it will only allow you to replay from the most recent crew run.

> Winner: LangGraph and Crew AI both make it easy to replay with inbuilt capabilities.

## Code Execution

Code execution capabilities enable agents to perform complex tasks by writing and executing code. This is particularly useful for tasks that require dynamic calculations or interactions with external systems.

LangGraph: LangGraph supports code execution through its integration with LangChain. Users can define nodes that execute code as part of the workflow.

Autogen: Autogen supports code execution through its built-in code executors. Agents can write and execute code to perform tasks autonomously. The framework provides a safe environment for code execution, ensuring that agents can perform tasks securely.

Crew AI: Crew AI supports code execution through customizable tools. Users can define tools that execute code and integrate them into the agent's workflow. This provides flexibility in defining the capabilities of agents and allows for dynamic task execution.

> Winner: Autogen might have a slight edge due to its built-in code executors but the other two are also capable.

## Human in the Loop

Human-in-the-loop interactions allow agents to receive human guidance and feedback, improving their performance and reliability. This is particularly important for tasks that require human judgment or intervention.

LangGraph: LangGraph supports human-in-the-loop interactions through its interruption features. Users can pause the graph execution to provide feedback or make adjustments.

Autogen: Autogen supports human-in-the-loop interactions through its 3 modes - NEVER, TERMINATE and ALWAYS.

Crew AI: Crew AI supports human-in-the-loop interactions by allowing agents to request human input during task execution by setting the human_input flag in the task definition. When enabled, the agent prompts the user for input before delivering its final answer.

Winner: All frameworks support humans in the loop in different ways.

## Customization

Customization options determine how easily developers can tailor the framework to their specific needs and requirements. This includes the ability to define custom workflows, tools, and interactions.

LangGraph: LangGraph provides fine-grained control over the flow and state of the application. Users can customize the behavior of nodes and edges to suit their specific needs. The framework's graph-based approach makes it easy to define complex workflows.

Autogen: Autogen is customizable, allowing users to extend agents with additional components and define custom workflows. The framework is designed to be modular and easy to maintain.

Crew AI: Crew AI offers extensive customization options, including role-based agent design and customizable tools.

Winner: All the frameworks provide customization but the mileage might vary

**Table of contents** Show

## Scalability

Scalability is a must to ensure that the framework can grow alongside your requirements. As you incorporate more agents, tools, and interactions, the framework should sustain its performance and reliability. All the three frameworks offer the flexibility to scale the system by adding agents, tools, and customizations according to your needs.

> Winner: It remains unclear which framework scales more effectively as more elements are added. We recommend experimenting with them to get a better idea.

## Comparison Summary

Woah, that was a lot of information! Here's a quick summary to make it easier to digest.

| Criteria | LangGraph | Autogen | Crew AI | Final Verdict |
|---|---|---|---|---|
| Ease of Usage | ✗ | ✓ | ✓ | Autogen and Crew AI are more intuitive due to their conversational approach and simplicity. |
| Multi-Agent Support | ✓ | ✓ | ✓ | Crew AI excels with its structured role-based design and efficient interaction management among multiple agents. |
| Tool Coverage | ✓ | ✓ | ✓ | LangGraph and Crew AI have a slight edge due to their extensive integration with LangChain. |
| Memory Support | ✓ | ✓ | ✓ | LangGraph and Crew AI are advanced in memory support features, ensuring contextual awareness and learning over time. |
| Structured Output | ✓ | ✓ | ✓ | LangGraph and Crew AI have strong support for structured outputs that are versatile and integrable. |
| Documentation | ✓ | ✓ | ✓ | LangGraph and Crew AI offer extensive and well-structured documentation, making it easier to get started and find examples. |

| | | | | |
|---|---|---|---|---|
| Multi-Agent Pattern Support | ✓ | ✓ | ✓ | LangGraph stands out due to its graph-based approach which makes it easier to visualize and manage complex interactions. |
| Caching | ✓ | ✓ | ✓ | LangGraph and Crew AI lead with comprehensive caching mechanisms that enhance performance. |
| Replay | ✓ | ✗ | ✓ | LangGraph and Crew AI have inbuilt replay functionalities, making them suitable for thorough debugging. |
| Code Execution | ✓ | ✓ | ✓ | Autogen takes the lead slightly with its innate code executors but others are also capable. |
| Human in the Loop | ✓ | ✓ | ✓ | All frameworks provide effective human interaction support and hence, are equally strong in this criterion. |
| Customization | ✓ | ✓ | ✓ | All the frameworks offer high levels of customization, serving various requirements effectively. |
| Scalability | ✓ | ✓ | ✓ | All frameworks are capable of scaling effectively, recommend experimenting with each to understand the best fit. |
| Open source LLMs | ✓ | ✓ | ✓ | All frameworks support open source LLMs. |

## Conclusion

We hope this blog clarifies the state of the top agent frameworks. LangGraph excels in scenarios where workflows can be represented as graphs, Autogen is ideal for conversational workflows, and Crew AI is designed for role-based multi-agent interactions. By understanding the key aspects of each framework, you can select the one that best aligns with your requirements.

Galileo's GenAI Studio makes AI agent evaluation a whole lot faster. Try GenAI Studio for yourself today!

Galileo

Subscribe to our newsletter

Subscribe

| RESEARCH | MODULES | RESOURCES | COMPANY |
|---|---|---|---|
| Evaluation Efficiency | Evaluate | Docs | Careers |
| Hallucination Detection | Observe | Blog | Privacy Policy |

AI System Diagnostics
High Quality Fine-Tuning
Powerful Metrics

Protect
Guardrail Metric

Request a Demo

Contact us at
© 2024 Galileo. All rights reserved.