ARTIFICIAL INTELLIGENCE

AI Agent Workflows: A Complete Guide on Whether to Build LangGraph or LangChain

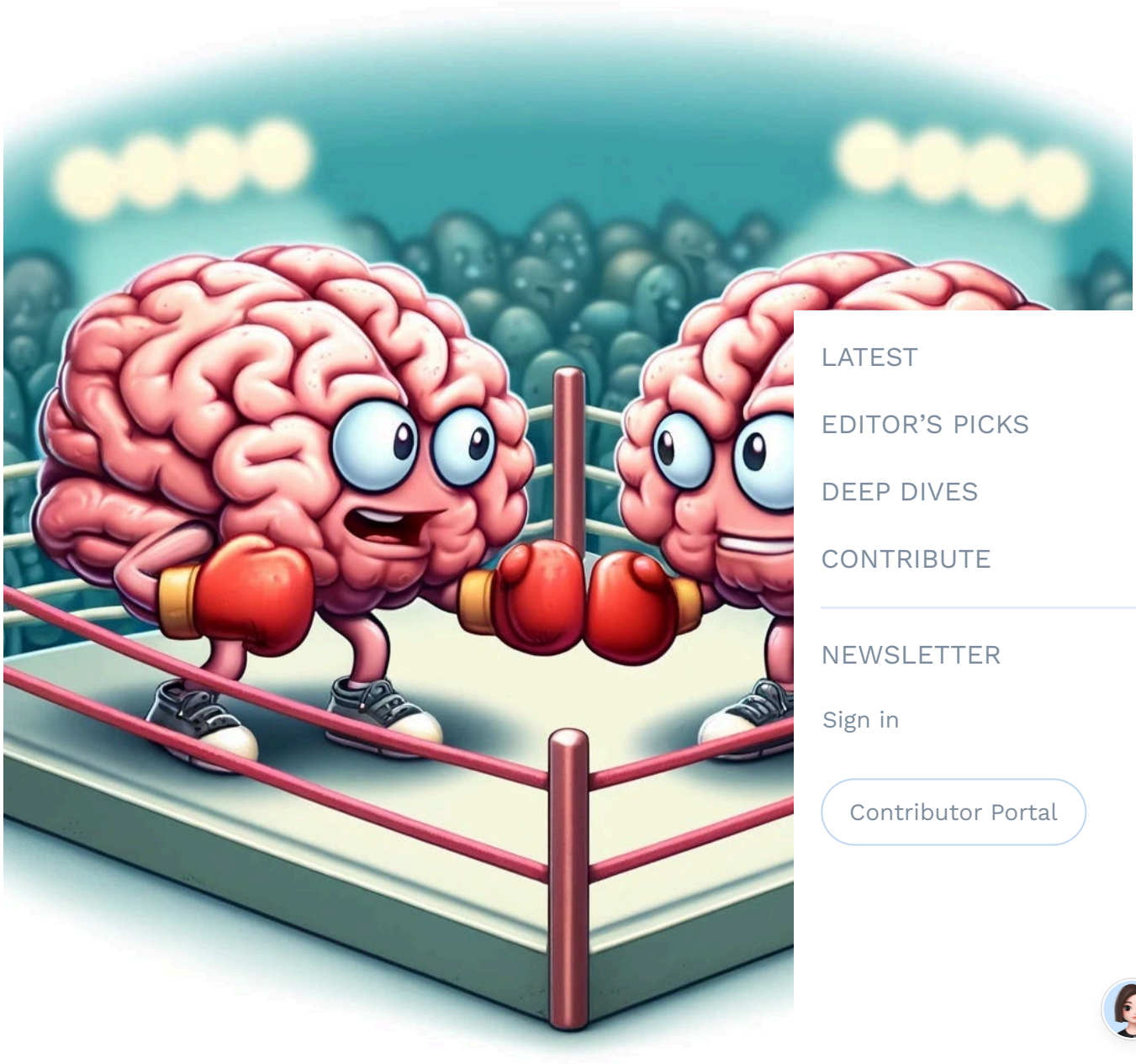
LangChain and LangGraph – A deep dive on key build
handle core pieces of their functionality, and decidin

Sandi Besen
Oct 25, 2024 12 min read

**A deep dive into two libraries by the same creat
LangGraph: their key building blocks, how they l
of their functionality, and deciding between the**

Language models have unlocked possibilities fo
interact with AI systems and how these systems
with each other – through natural language.

> When enterprises want to build solutions usi
> capabilities one of the first technical questio
> tools do I use?" For those that are eager to g
> first roadblock.



Source:Dalle-3

In this article, we will explore two of the most p for building Agentic AI Applications – LangChair the end of this article you should have a thorou the key building blocks, how each framework di core pieces of their functionality, and be able to point of view on which framework best fits your

Since the practice of widely incorporating Gener solutions is relatively new, open-source players competing to develop the "best" agent framewo

tools. This means that although each player brings a unique approach to the table, they are rolling out new functionality near constantly. When reading this piece keep in mind that what's true today, might not be true tomorrow!

Note: I originally intended to draw the comparison between AutoGen, LangChain, and LangGraph. However, AutoGen has announced that it launching AutoGen 0.4, a complete redesign of t the foundation up. Look out for another article w launches!

Base Components Of LangChain and

By understanding the different base elements o you will have a richer understanding of the key c they handle certain core functionality in the nex description is not an exhaustive list of all of the each framework, but serves as a strong basis to difference in their general approach.

LangChain

There are two methods for working with LangCh chain of predefined commands or using LangCh approach is different in the way it handles tools A chain follows a predefined linear workflow wh a coordinator that can make more dynamic (nor

- Chains: A sequence of steps that can includ agent, tool, external data source, procedura Chains can branch, meaning a single chain t paths based on logical conditions.

- Agents or Language Models: A Language Model has the ability to generate responses in natural language. But the Agent uses a language model plus added capabilities to reason, call tools, and repeat the process of calling tools in case there are any failures.
- Tools: Code based functions that can be called in the chain or invoked by an agent to interact with externa
- Prompts: This can include a system prompt model how to complete a task and what too information injected from external data sou the model more context, and the user prom model to complete.

LangGraph

LangGraph approaches AI workflows from a diff Much like the name suggests, it orchestrates wo graph. Because of its flexibility in handling differ AI agents, procedural code, and other tools, it is use cases where a linear chain method, branche agent system wouldn't meet the needs of the us was designed to handle more complex conditior feedback loops compared to LangChain.

- Graphs: A flexible way of organizing a workfl calls to an Llm, tool, external data source, p more. LangGraph supports cyclical graphs a you can create loops and feedback mechan be revisited multiple times.
- Nodes: Represent steps in the workflow, su an API call, or tool execution.

- Edges and Conditional Edges: Edges define the flow of information by connecting the output of one node as the input to the next. A conditional edge defines the flow of information from one node to another if a certain condition is met. Developers can custom define these conditions.
- State: State is the current status of the application as information flows through the graph. It is a mutable TypedDict object that contains all t information for the current execution of the automatically handles the updating of state information flows through the graph.
- Agents or Language Models: Language mode are solely responsible for generating a text r input. The agent capability leverages a langu enables the graph to have multiple nodes re components of the agent (such as reasoning execution of a tool). The agent can make de path to take in the graph, update the state perform more tasks than just text generatio

The Difference Between How Each F Handles Core Functionality

LangGraph and LangChain overlap in some of th they approach the problem from a different per focuses on either linear workflows through the different AI agent patterns. While LangGraph foc more flexible, granular, process based workflow agents, tool calls, procedural code, and more.

In general, LangChain require less of a learning LangGraph. There are more abstractions and pre

configurations that make LangChain easier to implement for simple use cases. LangGraph allows more custom control over the design of the workflow, which means that it is less abstracted and the developer needs to learn more to use the framework effectively.

Tool Calling:

LangChain

In LangChain there are two ways tools can be ca
if you are using a chain to sequence a series of
using its agent capabilities without it being expl
chain. In a chain, tools are included as a pre-de
chain – meaning that they aren't necessarily cal
because it was already predetermined they were
in the chain. However, when you have an agent
chain, the agent has autonomy to decided what
when based on the list of tools it is privy to.

Example of Flow for a Chain:

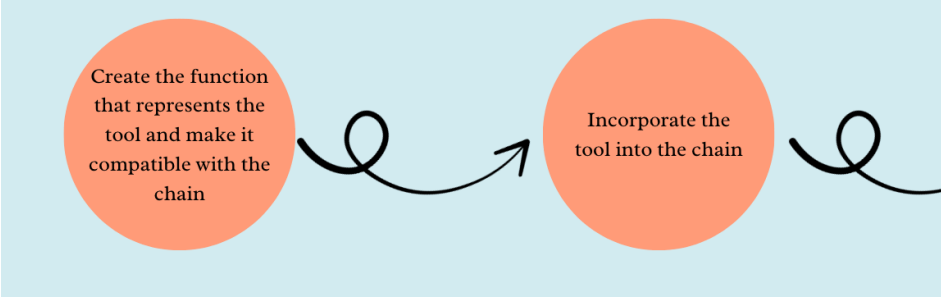


Image by Sandi Besen

1. Create the function that represents the tool
   compatible with the chain
2. Incorporate the tool into the chain
3. Execute the chain

Example of Flow for an Agent :

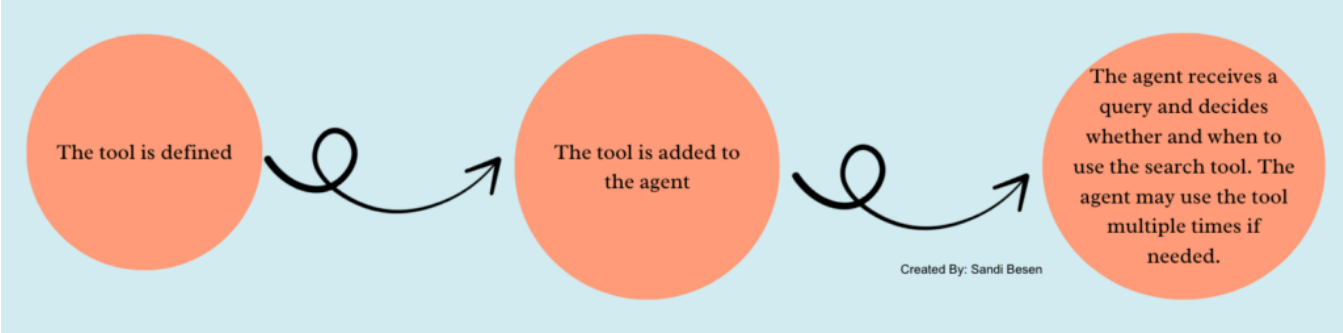


Image by Sandi Besen

1. The tool is defined
2. The tool is added to the agent
3. The agent receives a query and decides whe
   use the search tool. The agent may use the
   if needed.

LangGraph

In **LangGraph**, tools are usually represented as a
If the graph contains an agent, then then it is th
determines which tool to invoke based on its re
Based on the agent's tool decision, the graph na
node" to handle the execution of the tool. Cond
included in the edge from the agent to the tool
additional logic that determines if a tool gets ex
the developer another layer of control if desired
in the graph, then much like in LanchChain's cha
included in the workflow based on conditional l

Example of Flow for a Graph with anAgent:

LATEST
EDITOR'S PICKS
DEEP DIVES
CONTRIBUTE
NEWSLETTER
Sign in
Contributor Portal

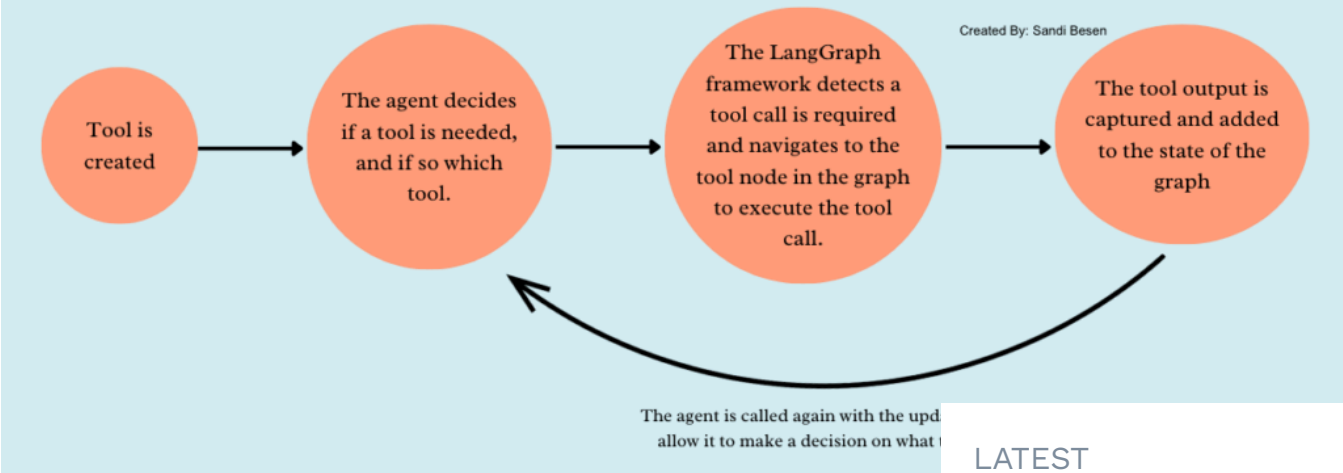


Image by Sandi Besen

1. The tool is defined
2. the tool is bound to the agent
3. The agent decides if a tool is needed, and if
4. The LangGraph framework detects a tool ca navigates to the tool node in the graph to e
5. The tool output is captured and added to th
6. The agent is called again with the updated make a decision on what to do next

Example of Flow for a graph without an Agent:

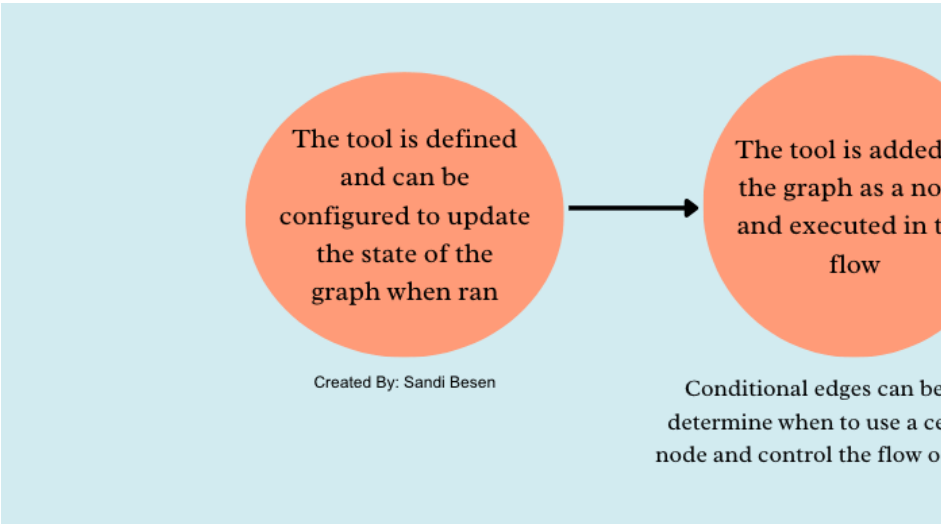


Image by Sandi Besen

1. The tool is defined

LATEST
EDITOR'S PICKS
DEEP DIVES
CONTRIBUTE
NEWSLETTER
Sign in
Contributor Portal

2. The tool is added to the graph as a node
3. Conditional edges can be used to determine when to use a certain tool node and control the flow of the graph
4. The tool can be configured to update the state of the graph

If you want to learn more about tool calling, my friend Tula Masterman has an excellent article about how to Generative AI.

Note: Neither LangChain nor LangGraph support out of the box like MSFT's Semantic Kernel.

Conversation History and Memory

LangChain

Langchain offers built-in abstractions for handli history and memory. There are options for the le (and therefore the amount of tokens) you'd like which include the full session conversation hist version, or a custom defined memory. Develope custom long term memory systems where they in external databases to be retrieved when relev

LangGraph

In LangGraph, the state handles memory by kee defined variables at every point in time. State ca like conversation history, steps of a plan, the ou model's previous response, and more. It can be node to the next so that each node has access state of the system is. However, long term persi across sessions is not available as a direct featu

LATEST
EDITOR'S PICKS
DEEP DIVES
CONTRIBUTE
NEWSLETTER
Sign in
Contributor Portal

framework. To implement this, developers could include nodes responsible to store memories and other variables in an external database to be retrieved later.

Out of the box RAG capabilities:

LangChain

LangChain can handle complex retrieval and ger
and has a more established set of tools to help
integrate RAG into their application. For instanc
document loading, text parsing, embedding crea
and retrieval capabilities out of the box by using
langchain.document_loaders, langchain.embedd
langchain.vectorstores directly.

LangGraph

In LangGraph, RAG needs to be developed from
the graph structure. For example there could be
document parsing, embedding, and retrieval tha
connected by normal or conditional edges. The s
would be used to pass information between ste
pipeline.

Parallelism:

LangChain

LangChain offers the opportunity to run multipl
parallel by using the RunnableParallel class. For
parallel processing and asynchronous tool callin
would have to custom implement these capabil
python libraries such as ayncio.

LangGraph

LangGraph supports the parallel execution of nodes, as long as there aren't any dependencies (like the output of one language model's response as an input for the next node). This means that it can support multiple agents running at the same time in a graph as long as they are not dependent nodes. Like LangChain, LangGraph can use a RunnableParallel class to run multiple
LangGraph also supports parallel tool calling by
libraries like ayncio.

Retry Logic and Error Handling:

LangChain

In LangChain, the error handling is explicitly def
developer and can either be done by introducing
chain its self or in the agent if a tool call fails.

LangGraph

In LangGraph you can actually embed error han
workflow by having it be its own node. When ce
can point to another node or have the same noc
part is that only the particular node that fails is
entire workflow. This means the graph can resu
of failure rather than having to start from the be
case requires many steps and tool calls, this co

In Summary

You can use LangChain without LangGraph, Lang
LangChain, or both together! It's also completely
using LangGraph's graph based orchestration wi

frameworks like MSFT's AutoGen by making the AutoGen Agents their own nodes in the graph. Safe to say there are a lot of option – and it can feel overwhelming.

So after all this research, when should I use each? Although there are no hard and fast rules, below is my personal option:

Use LangChain Only When:

You need to quickly prototype or develop AI wor involve sequential tasks (such as such as docun generation, or summarization) that follow a prec pattern. Or you want to leverage AI agent patter dynamically make decisions, but you don't need over a complex workflow.

Use LangGraph Only When:

Your use case requires non-linear workflows wh components interact dynamically such as workf on conditions, need complex branching logic, er parallelism. You are willing to build custom impl components that are not abstracted for you like

Using LangChain and LanGraph Together When:

You enjoy the pre-built extracted components o the out of the box RAG capabilities, memory fur also want to manage complex task flows using l linear orchestration. Using both frameworks tog powerful tool for extracting the best abilities frc

Ultimately, whether you choose LangChain, Lang combination of both depends on the specific ne

Note: The opinions expressed both in this article and paper are solely those of the authors and do not necessarily reflect the views or policies of their respective employers.

Still have questions or think that something needs to be further clarified? Drop me a DM on LinkedIn! I'm always eager to engage in food for thought and iterate on my work.

References:

Navigating the New Types of LLM Agents and

Home

LangGraph and Research Agents | Pinecone

Chaining the Future: An In-depth Dive into La

From Basics to Advanced: Exploring LangGra

WRITTEN BY

Sandi Besen

See all from Sandi Besen

Topics:

Ai Agent | Artificial Intelligence | Generative Ai To