LangChain and LangGraph are both open source frameworks designed to help developers build applications with large language models.
So what are the differences and why use one over the other?
Well, I think a good place to start. Is to define what these two things are, and let's begin with LangChain.

Now, we've done a dedicated video on LangChain, and my guess is there's probably a pop up somewhere over my head right now
encouraging you to watch that video.
But don't click, not yet. Give me a moment to summarize LangChain.
Then at the end of this video, if you want any more, you can go back and check that one out.
Okay. Now, at its core, LangChain is a way for building LLM powered applications by executing a sequence of functions in a chain.
So let's say we're building an application and the first thing it needs to do is it needs to retrieve some data from a website.
Once we've done that, we move on to stage two,
And stage two is to summarize that data that we retrieved.
And then finally, we're going to use this summary to do something,
specifically, We're going to have it answer user questions.
So the workflow here is retrieve, summarize, and answer
Now we can use the LangChain to help us do this.
So let's start with the retrieve component.
Now, the retrieve component might consist of a LangChain component called a document loader.
Now a document loader is used to fetch and load content from various data sources,
and if some of those documents are large, we might choose to use a text splitter,
which is another LangChain component to split takes up into smaller, semantically meaningful chunks.
Okay, that's retrieve.
Now To summarize, we would use a chain, and the chain will orchestrate the summarization process.
Now, that might include constructing a prompt component to instruct an LLM to perform the summarization.
And that might also contain an LLM component to pass that request to the large language model of our choosing.
Okay, and then answer.
Well, we would build another chain and this chain might include a memory component,
so this is another component of LangChain, and that's used to store conversation, history and context,
And we'd throw in another prompt component and another LLM component to generate the answer based on the summary and the record context.
And the cool thing here is that the LLM that we use for the answer component,
may be a completely different large language model to the one we use in the summarize component.
LangChain modular architecture let's build complex workflows by combining these high level components.

Okay, now let's introduce LangGraph.
LangGraph is a specialized library within the LangChain ecosystem,
specifically designed for building stateful multi-agent systems that can handle complex nonlinear workflows.
So let's consider a task management assistant agent.
Now, the workflow here involves processing user input.
So let's start there, Process inputs,
And then to this workflow we are going to allow to add tasks, we're going to be able to complete tasks, and we are also going to be able to summarize task.
So this is the kind of the architecture of what we're trying to build here.
Now, LangGraph helps us create this as a graph structure,
where each one of these actions is considered as a node.
So at tasks, complete tasks, summarize, they're all nodes.
And then the transitions between these things, that's known as edges.
Now the central mode is the process input node.
So that's where the user input comes in,
and that's going to use an LLM component to understand the user intent and to route to the appropriate action node.
Now there's another component here that's quite central to this called state, the state component.
And the state component is used to maintain the task list across all the interaction.
So the adds task node adds new tasks to the state,
the complete task node marks tasks as finished,
and then the summarize node uses an LLM to generate an overview of current tasks.
All nodes can access and modify the state allowing for contextual stateful interactions.
The graph structure allows the assistant to handle various user requests in any order,
always returning back to the process input node after the action is complete.
LangGraph Architecture lets us create flexible stateful agents that can maintain context over extended interactions.

So let's directly compare, LangChain and LangGraph across a number of dimensions.
Let's start with the primary focus.
Now the Primary focus of LangGraph Is to create and manage what is known as multi agent systems and workflows.
The focus of LangChain is to provide an abstraction layer for chaining LLM operations into large language model applications.
That's the difference between the two.
Now, as for. Structure, LangChain adopts no surprise here a chain structure.
And that acts as a Dag,
that is an acronym for directed acyclic graph,
Which means that tasks are executed in a specific order always moving forward.
So, for example, we start with task number one then we'd have a branch for maybe tasks number two and task three,
and then we'd come back to the central task number four.
And this process is great where you know the exact sequence of steps that are needed.
Now, LangGraph's graph structure, on the other hand, is a little bit different because it allows for loops and revisiting previous states.
So we might have state A which can go backwards and forwards with state B and State C,

and this is beneficial for interactive systems where the next step might depend on evolving conditions or user input.
Now, when it comes to components, LangChain uses a bunch and we've mentioned many of these already,
that includes memory, There's the prompt component as well, There's also the LLM component,
which is how we actually pass things to the large language model,
And there's the agent component as well that forms, chains between all of these things.
Now, LangGraph uses a bunch of different components.
So we have nodes, we also have edges, And we have states,
and these are all part of a graph.
And speaking of state. That brings us nicely to state management.
I don't think we can say that LangChain has somewhat limited state management capabilities.
It can pass information forth through the chain,
but it doesn't easily maintain a persistent state across multiple runs.
That said, LangChain does have these memory components that can maintain some state across interactions.
LangGraph's state management, I'm going to say, is more robust,
And that's because state is a core component that all nodes can access and modify,
allowing for more complex, context aware behaviors.
Let's say use cases.
Well, LangChain really excels, particularly at sequential tasks like a process that retrieves data and then processes it and then outputs a result.
Again, that said, LangChain is able to handle non sequential tasks to some extent with its own agents feature,
but LangGraph's wheelhouse that really is. Scenarios that have a much more complex nature to them.
Complex systems requiring ongoing interaction and adaptation.
For example, a virtual assistant that needs to maintain context over long conversations and handle varying types of requests.
So that LangChain and LangGraph, two powerful frameworks for building applications that make use of large language models.
All right. That's all I got.
You can go watch that LangChain video now.
- Generated with https://kome.ai