Chaining and Graph

Friday, 25 April 2025 10:33 AM

Agent: agent is essentially a control flow that an LLM decides where to go.
Chaining:

we can introduce the concept of chaining.
And this is basically to take one output of a LM and giving it as an input to another LM.
So to compose one call over the other.
And here you can see an example of a rack flow retrieval augmentation generation where we give the first
LM our original question.
We then integrate embeddings.
So we take the question, we embed it and we retrieve with similar relevant documents which are probably
going to help answer that question.
And then we take the original prompt, we augment it and we send everything to the LM which finally generates the answer.

Chain vs Agent: Chain is one directional left to right, where as in agent its bi-directional with cycles

Router:

And an LLM router is a kind of chain which leverages the LLM to decide where do we want to go.
So we can use an LLM to decide whether we're going to execute code in branch one or code in branch two.
So it can be for example, to go and search in a database or to go and search over the web.
And the LLM can decide where to go.
So we're using the reasoning power of an LLM here.
Here for the first time, the LLM decides which tapes do we need to take?
And this gives us even more flexibility of building those kinds of systems.

These things can be achieved by lang chain, but if we want everything to be decided by LLM then we go for lang graph

Function calls:
Function calling is a very cool feature of certain llms where Besides our query that we send the LLM the question, we can send a description of tools.
So those are functions that we can execute in our back end and we can orchestrate them.
And we send the LLM also the description of those functions, their arguments, their title, what they're supposed to do and what their return value is.
And we can do that very easily with the tool decorator and then the LLM, if it finds appropriate, it can tell us that we need to invoke these functions with those arguments.
So hopefully we'll go will invoke those function with those arguments.
And we'll get back the answer that we want.

Agent fundamental design:

So basically the algorithm for this type of agent is very simple.
So we start.
We then use the LM to decide whether we need to use a tool.
So for example, to make a call to an API or a call to a database and query there, and then decides whether we need to call this tool or not, then we go and call this tool with the arguments that the LM chose us to execute this tool, we get an answer, and then we feed back everything to the LM, which can decide whether to use another tool or whether to return the answer to the user.