Video Transcript

What are AI Agents?

2024 will be the year of AI agents. So what are AI agents? And to start explaining that,

we have to look at the various shifts that we're seeing in the field of generative AI.

And the first shift I would like to talk to you about is this move from monolithic models to compound AI systems. So models on their own are limited by the data they've been trained on.

So that impacts what they know about the world and what sort of tasks they can solve.

They are also hard to adapt. So you could tune a model, but it would take an investment in data, and in resources.

So let's take a concrete example to illustrate this point. I want to plan a vacation for this summer, and I want to know how many vacation days are at my disposal.

What I can do is take my query, feed that into a model that can generate a response.

I think we can all expect that this answer will be incorrect, because the model doesn't know who I am and does not have access to this sensitive information about me.

So models on their own could be useful for a number of tasks, as we've seen in other videos. So they can help with summarizing documents, they can help me with creating first drafts for emails and different reports I'm trying to do.

But the magic gets unlocked when I start building systems around the model and actually take the model and integrate them into the existing processes I have.

So if we were to design a system to solve this, I would have to give the model access to the database where my vacation data is stored.

So that same query would get fed into the language model.

The difference now is the model would be prompted to create a search query, and that would be a search query that can go into the database that I have. So that would go and fetch the information from the database, output an answer, and then that would go back into the model that can generate a sentence to answer, so, "Maya, you have ten days left in your vacation database."

So the answer that I would get here would be correct.

This is an example of a compound AI system, and it recognizes that certain problems are better solved when you apply the principles of system design.

So, what does that mean? By the term "system", you can understand there's multiple components. So, systems are inherently modular. I can have a model, I can choose between tuned models, large language models, image generation models,but also I have programmatic components that can come around it.

So I can have output verifiers. I can have programs that can that can take a query and then break it down to increase the chances of the answer being correct. I can combine that with searching databases. I can combine that with different tools.

So when we talking about a system approaches,I can break down what I desire my program to do and pick the right components to be able to solve that.

And this is inherently easier to solve for than tuning a model. So that makes this much faster and quicker to adapt.

Okay, so the example I use below, is an example of a compound AI system.

You also might be popular with retrieval augmented generation (RAG), which is one of the most popular and commonly used compound AI systems out there.

Most RAG systems and the example I use below are defined in a certain way.

So if I bring a very different query, let's ask about the weather in this example here.

It's going to fail because this the path that this program has to follow is to always search my vacation policy database.

And that has nothing to do with the weather. So when we say the path to answer a query, we are talking about something called the control logic of a program.

So compound AI systems, we said most of them have programmatic control logic.

So that was something that I defined myself as the human. Now let's talk about, where do agents come in?

One other way of controlling the logic of a compound AI system is to put a large language model in charge, and this is only possible because we're seeing tremendous improvements in the capabilities of reasoning of large language models.

So large language models, you can feed them complex problems and you can prompt them to break them down and come up with a plan on how to tackle it.

Another way to think about it is, on one end of the spectrum, I'm telling my system to think fast, act as programmed, and don't deviate from the instructions I've given you.

And on the other end of the spectrum, you're designing your system to think slow.

So, create a plan, attack each part of the plan, see where you get stuck, see if you need to readjust the plan. So I might give you a complex question, and if you would just give me the first answer that pops into your head, very likely the answer might be wrong, but you have higher chances of success if you break it down, understand where you need external help to solve some parts of the problem, and maybe take an afternoon to solve it.

And when we put a LLMs in charge of the logic, this is when we're talking about an agentic approach.

So let's break down the components of LLM agents.

The first capability is the ability to reason, which we talked about. So, this is putting the model at the core of how problems are being solved.

The model will be prompted to come up with a plan and to reason about each step of the process along the way.

Another capability of agents is the ability to act.

And this is done by external programs that are known in the industry as tools.

So tools are external pieces of the program, and the model can define when to call them and how to call them in order to best execute the solution to the question they've been asked.

So an example of a tool can be search, searching the web, searching a database at their disposal.

Another example can be a calculator to do some math.

This could be a piece of program code that maybe might manipulate the database.

This can also be another language model that maybe you're trying to do a translation task, and you want a model that can be able to do that.

And there's so many other possibilities of what can do here. So these can be APIs.

Basically any piece of external program you want to give your model access to.

Third capability, that is the ability to access memory.

And the term "memory" can mean a couple of things.

So we talked about the models thinking through the program, kind of how you think out loud when you're trying to solve through a problem.

So those inner logs can be stored and can be useful to retrieve at different points in time.

But also this could be the history of conversations that you as a human had when interacting with the agent.

And that would allow to make the experience much more personalized.

So the way of configuring agents, there's many are ways to approach it.

One of the more most popular ways of going about it is through something called ReACT, which, as you can tell by the name, combines the reasoning and act components of LLM agents.

So let's make this very concrete.

What happens when I configure a REACT agent?

You have your user query that gets fed into a model. So an alarm the alarm is given a prompt. So the instructions that's given is don't give me the first answer that pops to you.

Think slow planning your work. And then try to execute something tried to act. And when you want to act, you can define whether, if you want to use external tools to help you come up with the solution.

Once you get you call a tool and you get an answer. Maybe it gave you the wrong answer or it came up with an error.

You can observe that. So the alarm would observe.

The answer would determine if it does answer the question at hand, or whether it needs to iterate on the plan and tackle it differently up until I get to a final answer.

So let's go back and make this very concrete again. Let's talk about my vacation example. And as you can tell, I'm really excited to go on one, so I want to take the rest of my vacation days.

I'm planning to go on to Florida next month. I'm planning on being outdoors a lot and I'm prone to burning. So I want to know what is the number of two ounce sunscreen bottles that I should bring with me?

And this is a complex problem. So there's a first thing.

There's a number of things to plan. One is how many vacation days am I planning to take? And maybe that is information the system can retrieve from its memory. Because I asked that question before. Two is how many hours do I plan to be in the sun? I said, I plan to be in there a lot, so maybe that would mean looking into the weather forecast, for next month in Florida and seeing what is the average sun hours that are expected. Three is trying maybe going to a public health website to understand what is the recommended dosage of sunscreen per hour in the sun.

And then for doing some math, to be able to determine how much of that sunscreen fits into two ounce bottles. So that's quite complicated.

But what's really powerful here is there's so many paths that can be explored in order to solve a problem. So this makes the system quite modular.

And I can hit it with much more complex problems. So going back to the concept of compound AI systems, compound AI systems are here to stay. What we're going to observe this year is that they're going to become more agentic. The way I like to think about it is you have a sliding scale of AI autonomy. And you would the person defining the system would examine what tradeoffs they want in terms of autonomy in the system for certain problems, especially problems that are narrow, well-defined. So you don't expect someone to ask them about the whether when they need to ask about vacations.

So a narrow problem set.

You can define a narrow system like this one.

It's more efficient to go the programmatic route because every single query will be answered the same way.

If I were to apply the agentic approach here, there might be unnecessarily looping and iteration. So for narrow problems, pragmatic approach can be more efficient than going the generic route.

But if I expect to have a system, accomplish very complex tasks like, say, trying to solve GitHub issues independently, and handle a variety of queries, a spectrum of queries.

This is where an agentic route can be helpful, because it would take you too much effort to configure every single path in the system.

And we're still in the early days of agent systems.

We're seeing rapid progress when you combine the effects of system design with agentic behavior.

And of course, you will have a human in the loop in most cases as the accuracy is improving.

I hope you found this video very useful, and please subscribe to the channel to learn more.