1.7 Transcript

Hello, everyone. Welcome to today's session.

Today we're looking at chain of thought prompting and reasoning large language models to expand our topic.

First, let's look at the chain of thought prompting.

So it's a, like I said, chain of thought prompting is a prompting technique and it guides the model to think in steps before giving the final answer.

And it was first introduced by a paper from the Google Brain team.

And I have the source here. And to understand chain of thought prompting, let's actually compare it with standard prompting by looking at the example.

We have, uh, here for on the left hand side you have the standard prompting technique where you give the.

So essentially we're doing a one shot prompting. Right.

You have a sample question, you have the sample answer, and you have an actual question.

You ask the model. And we expect the model to output you an answer.

So in the input you have kind of uh example question answer format and another question and
also differ from the standard one shot prompting or few-shot prompting by adding your own, you know, thought process to the sample answer.

And in this, you know, simple example, we see, we add, uh, you know, the mathematical thought process to solve this simple math word problem.

And the model actually output its own thought process by mimicking our reasoning steps, but generating a chain of thought, a series of intermediate reasoning steps, we can significantly improve the ability of large language models to perform complex reasoning.

And as a prompting technique. It works like this. Right?

So you you're chatting into a large language model, and in your prompt you have your question, you have your chain of thought, and you have your answer. So this is, uh, organic unit.

Okay. That completes your...the question answer, uh, loop.

And then you ask another question and you expect the large language model to respond in a similar format.

So it should respond by providing its own chain of thought on the question and then provide you the final answer.

So the researchers found that.

This chain of thought technique improves multi-step reasoning, and it works best in large, larger language models in a sense that it's, uh, it elicits reasoning, and you need a sufficiently large language model to have the reasoning kind of capability in a book, and the chain of our prompting makes reasoning more interpretable because, you know, uh, you have the intermediate steps visible to you instead of you have a black box and a machine just outputting you the answer.

You can check your you know, how the machine arrive at the answer so you have more intermediate steps.

And this naturally leads to, you know, the new reasoning large language model.

And they utilize the chain method in a very clever way. Let me tell you about it.

And like the standard knowledge language models reasoning models, they generate a long internal chain of thought before producing an answer.

And this is all done automatically. So compared to the prompting technique you now have a new model.

So you know you don't need a prompt. Essentially the typical interaction is like this.

You ask a question and the reasoning. Large language models.

They were already trained to generate an internal chain of thought before outputting a final answer.

And they are trained to produce this, uh, richer intermediate reasoning steps by reinforcement learning.

So they were encouraged to think for a very long time before outputting an answer.

That being said, they're better at complex problem solving, coding, a scientific task, anything that you can think you know, reasoning is important and they excel in multi-step planning and logic.

Here is a typical multi round conversation look like in turn one you have your question.

You know train of thought and you have your answer. And then I'll show you how the context build up in turn two.

Right.

We abandon or essentially we discard the chain of content from turn one because, uh, we assume all the truth where all the power is in the answer.

So we collect question one and answer one, and that will be our new context.

And then we have the, you know, the question two, which is the new question for this turn.

And a new chain of thought is formulated before the final answer is returned.

In this way you build up your context and a new chain of thought is generated.

It's wrong and you don't really, you know, you don't save the chain thought from turn one.

I feed it to turn two because we assume all the information is already in the answer.

So that should be redundant. And now let's talk about the prompting advice.

So because there are essentially a different kind of model right.

Reasoning large language model versus generic large language model, that's better for uh, a chat or more simple task reasoning model.

Perform best with straightforward prompts and, uh, general structure.

You can think of this like this. You give them, uh, very clear and straightforward code, and you provide a lot of context.

So one example I'm using is, you know, help me plan a 15 minute online course on reasoning with large language models.

So essentially you want to see if I can replace me. Right.

And I give them more, uh, requirements such as I want to look at a deep speaker one and open it.

Oh one and I tell them my target audience.

And also another requirement is I want some cutting some code because we code in Python.

I'm being trained for some time and then I attach my knowledge.txt, which is uh, you know, I do this step because I order some material from OpenAI and DeepSeek.

I basically scrape their website and I just copy paste it.

Everything stored on here. This step is essential because I know both, uh, OpenAI-O1 and DeepSeek-R1 they have a knowledge cutoff sometime in 2023.

So the reasoning large language model concept might not be a thing.

They they all know chain of thought, but they. Uh, they might not even know themselves being introduced.

Let's see how the result is

So for the O-1 model, we have a summarized chain of thought available to us, and this is only available to us in the front end. In the back end API,

Uh, you can access this, but you can, you know, count the number of reasoning tokens.

We will talk about that later. So reviewing this chain of thought, you know, first it's a summarized chain of thought.

And um, but we can it's still very useful in the, in the sense that we can see how the model is thinking.

So it's first charting the course, uh, crafting the 15 minute course.

And, you know, drawing from the knowledge I gave them and formulating an outline for, for the course and then generating some code for us. Right.

Similarly, we have a different reasoning model, the DeepSeek-R1 for the for this model we actually have the complete chain-of-thought.

But if you have used it, you know that it's very chatty.

So the chain of thought can be very long and we won't read it here. And the point is you can access the complete chain of thought.

Right. And it's also available via the API.

You can check Notebook for more detail, but the chain of thought prompting is available to you and you can use it to understand your model better.

Or you can distill the knowledge in there.

Or you can also note can output that to your user if you are building your service around, uh, the DeepSeek-R1.

And then let's compare it there and the general structure of the response.

On the left hand side, we have the OpenAI's response. Uh, on our right hand side, we have the DeepSeek-R1 response.

And upon reviewing, uh,I have to say the, their structure don't differ that much.

And the content are mostly from the knowledge file, and they all reflect the most updated knowledge on reasoning large language model.

Several things I want to point out is, you know, we all have a overview or introduction section, and we all have a kind of key features for the the model and, you know, core concepts.

They're kind of essentially the same. And then let's move on to the coding parts generative.

So for the coding part now things are a bit different for for the O1.

Um, generative code. We see that it's, you know, first helping you to download all the packages and then.

Running a very simple example. And this is.

You know, almost identical to what I've provided in the example.

Uh, in the example knowledge txt file and our right hand side, we actually see some I won't say hallucination, but this is definitely, uh, misalignment because for I required some coding O1 but what I have is some coding O3 mini.

Right. And this keeps happening even though, you know, the general structure is right and we have a very capable reasoning capability, this can still happen sometimes.

For more code you can also review this. We have the DeepSeek-R1 code on the left hand side right.

This is generated by OpenAI. And we have more uh, code generated by DeepSeek-R1 on the right hand side.

And a similar issue happened with the O1 model being confused with O3

Now let's move on to the next section and review the conclusions from both models.

These are again essentially very similar as well.

We all have a best practice and wrap up session or like a best practice and Q&A section and like a reference or summary.

So, you know, these responses are all very structured.

If you remember you chatting with GPT-4, uh, sometimes you have a less structured answer.

These are turn out to be very structured for now. You know, we already saw the responses from the recent models.

And let's compare them with, uh, you know, responses from the chat models.

And here I will only give you a very high level summary. I will navigate you to the Notebook for the complete response because they are very long.

Right. And so I had observed some minor misalignment in the response from the chat model.

Specifically for the DeepSeek V3, it's uh, like the R1.

It also generated examples for the O3-mini instead of O1 and otherwise they are almost identical.

It turns out our 15-minute example is too simple for, you know, to show a big difference.

And a more practical use case would be to.

If you want to add the feature to a software that involves editing multiple files in a directory simultaneously, and that involves step by step thinking, right? So you plan out your, you know, the code you want to change.

And like, um, adding this feature will, you know, affect which code in my code base.

And then you execute those changes. Right. So that's when the reasoning capability really excels.

And similarly, when you are dealing with multi-step planning, when you know you can't use a reasoning model as the planner, producing a detailed, multi-step solution to a problem and then selecting and assigning a right chat model be it ChatGPT-4o or 4o mini.

And you can use those less capable models for, uh, the task for each step.

And now let's talk about the prompting best practice.

So because reasoning large language models are large language models, but they went through fine tuning, and they went through some reinforcement learning to enhance their reasoning capability.

They require some different prompting um, practice.

So number one is, uh, you should always keep your prompt straightforward and minimal.

Reasoning models are like your senior coworkers, right? They don't require a lot of instructions.

You don't need to spoon feed them everything, and you can always try a zero shot before full shot.

Like you can be trusted with the output, with figuring out how to, you know, output in the correct format.

And. You essentially need to provide a example in the first place.

And because these examples are in a training step, they're encouraged to output the chain of thought.

You don't need to manually put your training for prompt. The already trained to do so.

So prompts such as thinking step by step is unnecessary.

The third point is you be specific about your code and your output format.

So in the prompt I use, I just ask it to uh, design a course, for me. But for very specific tasks such as updating code in a code base.

You might be specific about, you know, I want to I want you to reflect all the changes in XML format, and I want all the changes to be, uh, the directory and the, you know, the contents changed so you actually can automate based on the output.

Okay. And to summarize reasoning large English models are very capable in your models and they generate internal US out before final answer.

They require a minimal prompting but they also benefit from specific output instruction.

And you can check the notebook for more code examples and API info.

And the key takeaways are reasoning models are very capable of new models.

And if you understand this capability and how to use it, you can, you know, take advantage of it to the fullest.

Thank you for taking today's class.