MY WEIRD PROMPTS

Podcast Transcript

EPISODE #181

Inside the Brain of an AI: The Rise of Reasoning Models

Published January 06, 2026 • Runtime: 31:23

https://myweirdprompts.com/episode/ai-reasoning-models-explained/

EPISODE SYNOPSIS

In this episode of My Weird Prompts, Corn and Herman Poppleberry dive deep into the seismic shift occurring in artificial intelligence: the transition from fast, predictive chatbots to slow, deliberate reasoning models. They explore the engineering behind "inference-time compute scaling," explaining how hidden tokens and "System 2" thinking allow models to catch their own errors before they even reach the user. By breaking down complex concepts like Monte Carlo Tree Search and Process Reward Models, the brothers reveal what happens when you crank an AI's "reasoning level" to the max and why the future of tech depends on an AI's ability to show its work. Whether you're a software engineer or just curious about the data center's rising energy costs, this deep dive explains why the most powerful AI isn't necessarily the biggest, but the one that thinks the longest.

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. It is a bit of a gray, rainy afternoon here in Jerusalem, but we are cozy inside and ready to dive into some serious tech. I am Corn, and sitting across from me is my brother, the man who once tried to explain quantum entanglement using only sourdough starter.

Herman

Herman Poppleberry, at your service. And honestly Corn, I think the rain is the perfect backdrop for today's topic. It feels like the kind of weather where you just want to sit down and really think through a complex problem. The kind of day where you don't want a quick answer; you want a deep, considered one. Which is fitting, because that is exactly where the world of artificial intelligence has moved over the last year.

Corn

Which is exactly what our housemate Daniel was asking about in the prompt he sent over this morning. He was looking at how much the landscape of artificial intelligence has shifted over the last twelve to eighteen months. If you remember, back in early twenty-twenty-five, we started seeing this massive shift toward what people are calling thinking models or reasoning models. We moved past the era of the chatbot that just blurted things out and into the era of the A-I that actually stops to ponder.

Herman

It has been the defining trend. We went from models that just predict the next word very quickly to models that actually pause, deliberate, and essentially talk to themselves before they give you an answer. It is the difference between a reflex and a reflection. And for Daniel, who is an engineer, the magic isn't in the interface—it is in the plumbing.

Corn

Right, and Daniel's question gets right to the heart of the engineering behind it. He wants to know what is actually happening under the hood when we enable these thinking modes. Especially that parameter we see in so many interfaces now where you can set the reasoning level to low, medium, or high, or even a sliding scale from zero to one hundred. What is the model actually being forced to do when you crank that up to the max? Is it just running the same code more times, or is something fundamentally different happening in the data center?

Herman

It is such a great question because it touches on a fundamental change in how we use compute power. For a long time, the goal was just to make models bigger during the training phase—more parameters, more data, more pre-training. But now, we are seeing the rise of what researchers call inference-time compute scaling. Basically, spending more energy and time while the model is actually answering the question, rather than just while it is being built. It is the realization that a smaller model that thinks for a minute can often outperform a massive model that thinks for a second.

Corn

I think a good place to start for our listeners is the distinction between a standard large language model and one of these thinking variants. To a casual user, they might look similar—you type a prompt, you get a response—but the internal architecture, or at least the process flow, is quite different, right?

Herman

Exactly. Think of a standard model like a person who is incredibly well-read but also very impulsive. If you ask them a question, they start talking immediately. They are relying on their intuition and the patterns they have seen in billions of pages of text. This is what psychologists call System One thinking. It is fast, instinctive, and often very effective for simple tasks like writing a poem or summarizing a news article.

Corn

But it is also prone to those classic hallucinations or logic errors because it has already committed to the start of the sentence before it has worked out the end of the logic. It is like that friend who starts a story without knowing the punchline and then has to make it up as they go.

Herman

Precisely. Now, a thinking model—like the OpenAI o-one series or the DeepSeek-R-one models that dominated twenty-twenty-five—is more like that same person if they were given a scratchpad and told they cannot speak until they have mapped out the entire logic of their answer. That is System Two thinking. It is slow, deliberate, and analytical. Under the hood, the biggest difference is that these models are trained using a process called reinforcement learning to generate a chain of thought.

Corn

And that chain of thought is often hidden from the user, though as Daniel mentioned, many platforms now let you toggle that view so you can see the model essentially arguing with itself. I've seen it happen—it will say something like, wait, that contradicts the second law of thermodynamics, let me try again. It's a bit eerie.

Herman

It is! When you use a standard model, if you ask it a difficult math problem, it might just guess the answer based on the most likely sequence of numbers. A thinking model, however, is trained to produce hidden tokens. These are words and symbols that the user usually does not see, where the model says things like, wait, if I carry the one here, then the next digit should be a four. It checks its own work as it goes. This is made possible by something called Process Reward Models, or P-R-Ms.

Corn

Process Reward Models. How do those differ from the way we used to train A-I?

Herman

In the old days—well, twenty-twenty-three—we used Outcome Reward Models. We gave the A-I a point if the final answer was right and zero points if it was wrong. But with thinking models, we give it rewards for every step of the reasoning process. We are essentially telling the A-I, I like how you double-checked that variable, or good job identifying that edge case. This encourages the model to develop these internal monologues that lead to better outcomes.

Corn

I remember when we were talking about the math of magic back in episode two hundred seventy-seven, we touched on how tensors and weights work. How does that connect here? Is the model actually changing its weights when it thinks? Does it get smarter the longer it sits there?

Herman

That is a common misconception. The weights of the model—the actual neural connections—are frozen during inference. They do not change while you are chatting with it. What changes is the context window. The model is using its own previous thoughts as input for its next thought. It is creating a feedback loop where it can refine its own logic. It is essentially building a bigger and bigger prompt for itself as it goes.

Corn

So, it is populating its memory with its own reasoning steps. Now, to get to Daniel's specific question about the parameters, like setting the thinking level to one hundred. From an engineering standpoint, what are we actually doing to the hardware when we hit that max button?

Herman

What we are doing there is controlling the search budget. That is the technical term. In computer science, we often use something called Monte Carlo Tree Search, or M-C-T-S. Imagine the problem is a giant tree with thousands of possible branches representing different logic paths. A low thinking setting tells the model to just look at a few branches and pick the most promising one quickly. It’s a shallow search.

Corn

And a high setting? If I crank it to one hundred, are we talking about a forest?

Herman

A high setting tells the model to explore hundreds or even thousands of branches. It might go down one path, realize it leads to a logical contradiction, backtrack, and try a different path. When you set that parameter to the maximum, you are essentially telling the system to keep searching until it reaches a very high level of confidence or until it hits a hard limit of computational tokens. It is using a technique called Best-of-N sampling or tree-of-thought processing.

Corn

This explains why it takes so much longer. If I am asking for a complex piece of code and I set the reasoning to max, I might be waiting thirty seconds or a minute. During that time, the model is basically running a mini-simulation of the problem over and over. It's like it's playing a game of chess against itself to find the best move.

Herman

Exactly. It is generating thousands of internal tokens that you will never see. It might be writing a version of the code, running a mental check to see if it would throw an error, and then discarding it and starting over. This is why these models are so much better at coding. They can catch their own bugs before they ever show you the final product. But there is a physical cost to this. Every one of those hidden tokens requires a pass through the G-P-U. When you set it to max, you are essentially pinning those H-one-hundred or B-two-hundred chips at high usage for a much longer duration.

Corn

It is fascinating because it feels like we are moving away from the idea of A-I as just a database search and more toward A-I as a processing engine. But I wonder about the cost. If it is doing all this extra work, someone has to pay for those G-P-Us spinning in a data center somewhere. I've noticed that some of these thinking models are significantly more expensive per request.

Herman

Oh, the cost is significant. That is why we see a tiered pricing model for these thinking tokens. It is also why the parameters are there in the first place. You do not need a thinking model to tell you a joke or summarize a meeting. You would be wasting massive amounts of energy and money for a task that System One can handle in a fraction of a second. It's about matching the compute to the complexity.

Corn

Right, it is like using a supercomputer to calculate your grocery bill. It works, but it is overkill. Before we get deeper into the specific engineering of these search trees and the reinforcement learning from human feedback that makes them possible, let's take a quick break for our sponsors. We have a very... interesting one today. Barry: Are you tired of your brain feeling like a twenty-twenty-two era chatbot? Slow, glitchy, and prone to forgetting where you put your keys? You need the Neural-Nudge Headband. Our patented bio-resonant copper coils align your thoughts with the Earth's natural magnetic field, supposedly. Users report a thirty percent increase in their ability to remember things they just saw, and some even claim they can hear the thoughts of nearby pigeons. It is not science, it is better than science. It is Barry's Neural-Nudge. Side effects may include a metallic taste in the mouth, an intense craving for birdseed, and a sudden, inexplicable knowledge of the migratory patterns of the common starling. Neural-Nudge. BUY NOW! Use code WEIRD for a free bag of premium cracked corn.

Corn

Thanks, Barry. I am not sure I want to hear what the pigeons in Jerusalem are thinking. They probably just want my sandwich. And I definitely don't need a metallic taste in my mouth while I'm trying to eat it.

Herman

Given the pigeons around our house, I suspect their thoughts are mostly about tactical maneuvers for crumb acquisition. They probably have a higher search budget for snacks than most A-I models. Anyway, back to the thinking models.

Corn

So, Herman, we were talking about the search budget and how the model explores different branches of logic. I want to go back to the training part. How do you actually teach a model to think? You can't just give it a book on logic and expect it to work. We tried that in the eighties and nineties with symbolic A-I, and it was a bit of a disaster.

Herman

You are right. You can't. The breakthrough that really accelerated things in twenty-twenty-five was a more sophisticated use of Reinforcement Learning, or R-L. Traditionally, we trained models to predict the next word in a sentence from a dataset. But for reasoning, researchers started using a process where the model is rewarded for the correct final answer, but also for the quality of its intermediate steps. This is often called R-L-A-I-F—Reinforcement Learning from A-I Feedback.

Corn

So, it is like a student who gets partial credit for showing their work, even if the final answer is slightly off? Or is it more about the path they took?

Herman

It’s more about the path. The most successful models are often rewarded primarily for the final answer being correct, which forces them to figure out on their own which thinking patterns lead to success. If the model thinks, I should use a loop here, and then it gets the answer right, the R-L algorithm strengthens the probability of that thinking pattern occurring again in the future. Over millions of iterations, the model discovers the rules of logic and math not because we programmed them in, but because those rules are the only way to consistently get the right answer. It’s emergent logic.

Corn

That is an important distinction. We aren't hard-coding logic into the A-I. We are creating an environment where logic is the most successful survival strategy for the model to get its reward. It’s like evolution, but for math problems.

Herman

Exactly. And when you adjust that thinking parameter Daniel was talking about, you are essentially letting the model use more of those learned logic patterns. When the parameter is low, the model might only take the top two or three most likely steps. When it is high, it explores the tail of the distribution, looking for those less obvious but more accurate connections. This is where the engineering gets tricky, because you have to manage the K-V cache—the memory of all those previous tokens—very carefully so the model doesn't run out of memory while it's overthinking.

Corn

I want to talk about the engineering of the back end when we hit that max button. In a standard inference setup, you have a request and a response. But with these reasoning models, there is almost a conversation happening between different parts of the system, right? Is there a supervisor model involved?

Herman

Often, yes. There is a supervisor model or a set of heuristics that monitors the chain of thought. If the model starts looping—saying the same thing over and over—or going down a completely nonsensical path, the system can intervene and say, stop, try a different branch. This is part of the inference-time compute scaling. We are essentially building a small software stack around the model to manage its thinking process. It’s not just one big neural net anymore; it’s a system.

Corn

So, when the parameter is at one hundred, is the model actually running multiple versions of itself in parallel? Like a committee of Hermans all arguing about the best way to explain a tensor?

Herman

It can be! There are different ways to implement it. One way is called Best-of-N sampling. You ask the model the same question ten times with a bit of randomness, and then a separate model or an automated checker picks the best result. But the more advanced thinking models, like the ones released in late twenty-twenty-five, do this internally. They use a single pass but with a massive number of tokens dedicated to searching the solution space. They use something called a verifier to check the logic of each step before moving to the next.

Corn

It makes me think of the episode we did on the gold standard of uptime, episode two hundred seventy-four. We talked about redundancy and checking. In a way, thinking models are building redundancy into their own logic. They are checking themselves so the user doesn't have to. But does this mean we can finally trust them? Are hallucinations a thing of the past?

Herman

They aren't gone, but they are much rarer. In early twenty-twenty-five, we saw the first models that could solve complex coding benchmarks that had stumped A-I for years—things like the LiveCodeBench or the A-I-M-O math competitions. The reason they succeeded wasn't that they were smarter in the traditional sense, but that they were allowed to think for five minutes instead of five seconds. However, they can still hallucinate in their thinking process. Sometimes the model will convince itself of a lie during its chain of thought and then follow that lie to a very confident, very wrong conclusion.

Corn

I've noticed this when I'm using these tools for my own projects. If I give it a simple script to write, it's done instantly. But if I ask it to refactor a whole library and I turn on the thinking mode, I can actually see it going through the files, identifying dependencies, and essentially planning the architecture before it writes a single line of code. It's like watching a master builder look at a blueprint.

Herman

And that planning is the key. In the old models, if the A-I made a mistake in the third line of a hundred-line file, the whole thing was ruined because it couldn't go back and fix it. With reasoning models, they catch that mistake in their internal draft and fix it before you ever see it. This is why the engineering community has pivoted so hard toward these models. We are moving from prompt engineering to flow engineering—designing the systems that allow these models to think most effectively.

Corn

So, if we look at the difference between a thinking variant and a non-thinking one, is the non-thinking one just the thinking one with the chain of thought turned off? Or are they actually different models? If I buy the cheap version, am I just getting a lobotomized genius?

Herman

Usually, they are the same base model, but the thinking variant has been fine-tuned on that specific reasoning data. It is like a person who has been trained in the Socratic method versus someone who hasn't. The base intelligence might be the same, but the trained person has a specific set of mental tools they can deploy. However, some companies do release smaller, faster versions of their models—distilled versions—that are stripped of the reasoning capabilities to save on costs and latency. They take the knowledge of the big model but remove the heavy thinking machinery.

Corn

That makes sense. You have your high-speed commuter car for daily tasks and your heavy-duty construction vehicle for the big jobs. But let's talk about the edge cases. What happens when you force a model to think too much about something simple? Does it start overthinking like a human does? Does it get existential about a sandwich recipe?

Herman

It actually does! This is one of the funniest things we have seen in the research over the last year. If you ask a max-reasoning model a very simple trick question, like, how many sisters does a boy have if he has three brothers, it might over-analyze it. It might start wondering if there are hidden cultural contexts, or if the question is a logic trap, or if the brothers are half-brothers. It can actually talk itself out of the correct, simple answer. It’s the A-I equivalent of a PhD student failing a kindergarten quiz because they’re looking for a deeper meaning that isn’t there.

Corn

I have seen that! It starts writing a three-paragraph essay on the definition of a family unit when all I wanted was a number. It is the classic problem of having a hammer and seeing everything as a nail. If you give a model a huge search budget, it feels obligated to spend it.

Herman

Exactly. That is why the parameter control Daniel mentioned is so important. It gives the human in the loop the ability to say, look, I just need a quick answer, don't overthink this. Or, conversely, this is a mission-critical piece of security code, spend as much time as you need to get it perfect. From a back-end standpoint, when you set that parameter to max, the system might also be increasing the temperature—the randomness—of the internal thoughts to encourage more creative problem-solving, which it then filters through that verifier we talked about.

Corn

Let's talk about the tokens for a second. When we talk about the max parameter, does that correlate directly to a token limit? Is there a hard cap on how many hidden thoughts an A-I can have before it just has to give up?

Herman

In most implementations, yes. Every word the model thinks or says is a token. When you set the reasoning to max, you are essentially raising the ceiling on how many hidden tokens the model is allowed to generate. On some of the high-end systems we are seeing here in early twenty-twenty-six, that limit can be in the tens of thousands. That is a lot of internal dialogue. Some models are now capable of thinking for twenty or thirty minutes on a single prompt. That is a massive amount of compute—literally thousands of times more than a standard chat response.

Corn

And that translates to real-world latency. I think that is the biggest hurdle for people to get over. We have spent the last five years wanting A-I to be faster, and now we are suddenly being told that slower is better. It’s a complete reversal of the tech industry’s move-fast-and-break-things mantra.

Herman

It is a shift in mindset. We are moving from A-I as a search engine to A-I as a collaborator. You wouldn't expect a human colleague to solve a complex architectural problem in three seconds. We are finally giving A-I the same grace period to actually work through the problem. And the engineering is catching up to support this—we have new types of memory management and specialized chips that are optimized for these long, sequential chains of thought rather than just parallel processing.

Corn

It is also interesting to see how this affects the open-source community. For a while, it seemed like only the giant companies with massive compute could do this. But then we had the rise of models like DeepSeek-R-one and the open-source reasoning variants from Meta and Mistral. They showed you could achieve high-level reasoning with much more efficient training methods.

Herman

That was a huge turning point in twenty-twenty-five. It proved that the secret sauce isn't just more G-P-Us, but better data and better reinforcement learning strategies. It leveled the playing field and allowed smaller developers to build thinking models for specific niches, like medical diagnosis or legal research, where accuracy is far more important than speed. We’re seeing models that are tiny—maybe only seven billion parameters—but because they’ve been distilled from the thinking of much larger models, they can reason through logic puzzles that would have crushed a hundred-billion-parameter model two years ago.

Corn

You mentioned medical and legal. Those seem like the perfect use cases for maxed-out reasoning. You don't want a fast answer from your doctor; you want the right one. You want them to consider every possible interaction and edge case.

Herman

Exactly. Imagine a model that spends ten minutes analyzing every research paper ever written on a specific rare condition before it gives a recommendation. That is the power of scaling inference compute. We are essentially trading electricity for accuracy. And as we get better at it, the cost will come down, but the principle remains: some problems are just hard, and hard problems take time.

Corn

So, looking forward, do you think we will eventually reach a point where the model knows how much to think without us telling it? Like, it sees the question and decides for itself if it needs a System One or System Two approach? I don't want to have to slide a bar every time I ask for a recipe.

Herman

We are already starting to see that. Some of the newest A-P-Is have an auto setting for reasoning. The model does a quick initial pass to assess the complexity of the prompt. If it is simple, like what is the capital of France, it responds immediately. If it detects a logic puzzle, a complex coding request, or a request for a nuanced ethical analysis, it automatically triggers the reasoning engine. It’s like a car that automatically shifts gears based on the terrain.

Corn

That feels like the natural evolution. It makes the tech more invisible. You just get the best answer in the most efficient way possible. But I suspect there will always be a group of people who want to override the auto-pilot.

Herman

I agree. There will always be a place for that manual control, especially for power users like Daniel. Sometimes you want to force the model to be more rigorous than it thinks it needs to be, just to see if it uncovers something new. Or maybe you want to turn the thinking off entirely to save money on a task you know is simple. It’s about agency.

Corn

It is like double-checking a proof. You know the answer is likely correct, but you want to see every single step of the logic to be absolutely sure. And that brings us back to the hidden tokens. One of the big debates in twenty-twenty-six is whether companies should be required to show those thinking steps to the user. Some argue it is essential for transparency and safety—so we can see if the A-I is being deceptive—while others say it is proprietary information or that it might confuse the average user.

Herman

It’s a heated debate. Some companies are actually hiding the thoughts because they don't want other companies to use those thoughts to train their own models—a process called distillation. But from a safety perspective, being able to see the model's internal monologue is a huge win. We can see if it’s trying to manipulate us or if it’s following a biased line of reasoning before it ever outputs the final text.

Corn

I personally love seeing the thoughts. It makes the A-I feel less like a magic box and more like a very hard-working intern. It is reassuring to see it say, wait, that won't work, let me try this instead. It humanizes the machine in a weird way, even though it’s just math.

Herman

It definitely builds trust. When you see the model catch its own mistake, you feel much more confident in the final result. It also helps you learn. I have actually improved my own coding by watching how the reasoning models break down complex problems into smaller, manageable steps. It’s like having a tutor who shows their work.

Corn

That is a great point. It is a pedagogical tool as much as a productivity tool. We are learning how to think better by watching the machines learn how to think. It’s a bit of a mirror, isn’t it?

Herman

It is a beautiful feedback loop. And it really shows how far we have come from the early days of just predicting the next word in a sentence. We are building systems that don't just mimic human speech, but mimic human thought processes—the slow, messy, iterative process of getting to the truth.

Corn

Well, I think we have given Daniel a lot to chew on. From search budgets and Monte Carlo trees to reinforcement learning and System Two thinking, there is a lot going on under that hood. It’s not just a slider; it’s a fundamental shift in how we use silicon to solve problems.

Herman

There really is. It is an exciting time to be in this field. We are finally moving past the era of the confident hallucinator and into the era of the thoughtful analyst. It’s the difference between a guess and a guarantee.

Corn

I like that. The thoughtful analyst. It has a nice ring to it. Before we wrap up, I want to remind everyone that if you are enjoying these deep dives, please leave us a review on your podcast app or on Spotify. It really does help the show reach more curious minds like yours. We’re a small operation, and every star counts.

Herman

It really does. And a big thanks to Daniel for sending in this prompt. It was a fun one to unpack. If any of you listening have a weird prompt of your own—whether it’s about A-I, physics, or the secret lives of Jerusalem pigeons—you can always reach out to us through the contact form at myweirdprompts dot com.

Corn

You can also find our full archive of episodes there, including the ones we mentioned today like episode two hundred seventy-seven on tensors and episode two hundred seventy-four on internet uptime. There is a whole world of rabbit holes to explore. We’ve got over three hundred episodes now, so there’s plenty to keep you busy during the next rainy afternoon.

Herman

And don't forget, we are on Spotify and most other podcast platforms. Just search for My Weird Prompts. We’re the ones with the logo that looks like a brain wearing a party hat.

Corn

Alright, I think the rain is finally letting up a bit. The sky is turning that weird shade of purple it gets over the Judean hills. Maybe we can actually go for that walk now.

Herman

As long as we don't run into any of Barry's Neural-Nudge customers. I'm not ready for that level of magnetic alignment or birdseed cravings. I'll stick to my own unaligned thoughts, thank you very much.

Corn

Fair enough. Thanks for listening, everyone. This has been My Weird Prompts.

Herman

Until next time, keep asking the weird questions. And maybe think twice before you hit that max reasoning button on a sandwich recipe.

Corn

Bye for now!

Herman

Bye!