MY WEIRD PROMPTS

Podcast Transcript

EPISODE #257

AI That Evolves: Solving the Preference Problem

Published January 20, 2026 • Runtime: 25:28

https://myweirdprompts.com/episode/ai-continuous-learning-preferences/

EPISODE SYNOPSIS

In this episode, Herman and Corn tackle a frustration shared by many power users: why can’t our AI assistants stay updated with our evolving tastes in real-time? From the limitations of static training data to the "context rot" that plagues current recommendation systems, the duo breaks down the engineering hurdles of building a truly adaptive partner. They explore cutting-edge solutions like Test-Time Training (TTT), self-editing memory architectures like Letta, and the potential for nightly personal fine-tuning using LoRA. Whether you're tired of "amnesiac" LLMs or curious about the next frontier of personalization, this deep dive into the AI feedback loop offers a glimpse into a future where your model grows alongside you.

DANIEL'S PROMPT

Daniel

"I'd like to ask a question about a specific implementation in AI engineering that I've long thought would be brilliant if it could work, but I haven't figured out if it's actually possible. I've done several interesting projects with AI over the past year, from work-related tasks to experiments like creating an agentic UN with agents representing countries. However, one basic application I've struggled with is asking an AI agent to recommend content, like a Netflix show or a YouTube playlist. The challenge is that the agent needs to know your preferences, which involves context data. But if you just provide a description of your likes and dislikes, you tend to get the same recommendations repeatedly. To improve this, you need a mechanism for storing rejections and tracking what you've already watched. Factors like geo-restricted content also complicate things. The biggest issue is that AI models are frozen in time with a knowledge cutoff. While we can use search to provide current information, user feedback is harder to integrate. Currently, training or fine-tuning happens in cycles: you gather information, feed it back in a new training session, and then release it. Is there any technology or AI model that has a continuous training loop? Instead of this iterative process, I'm looking for something like a continuous deployment model for AI self-correction from a feedback loop. Does this exist, is it possible, and how can we bring it to reality so we can have an AI bot that perfectly understands our preferences without manual sifting?"

TRANSCRIPT

Corn

Hey everyone, welcome back to My Weird Prompts. I am Corn, and I am sitting here in our living room in Jerusalem with my brother, looking out at a surprisingly clear afternoon. You can actually see the Judean Hills today, which is a rare treat after the dust we had last week.

Herman

And I am Herman Poppleberry, ready to dive into some serious engineering questions. It is good to be here, Corn. The air is crisp, the coffee is strong, and the prompts are getting weirder by the day.

Corn

They really are. And we have a really interesting one today from our housemate Daniel. He has been busy lately, has he not? He was telling me about that agentic United Nations project he was working on, where he had different artificial intelligence agents representing different countries. It was absolute chaos in the kitchen for three days.

Herman

I remember that. He was trying to see if they could actually negotiate a climate treaty without human intervention. It was a fascinating experiment in game theory, even if the agent representing one of the smaller nations ended up trying to buy all the carbon credits with digital goats. I think it was a hallucination triggered by a specific cultural data set he used, but it really highlighted how these agents can go off the rails when they start optimizing for local rewards.

Corn

Right, he is always pushing the boundaries of what these systems can do. But today, he is asking about something that sounds simpler on the surface but is actually a massive technical hurdle in the world of artificial intelligence. He wants to know why he cannot just have a recommendation bot, like for Netflix or YouTube, that actually learns from him in real time and stays updated without him having to manually feed it his life story every single time. He is tired of the manual sifting.

Herman

It is the classic preference problem, Corn. Daniel pointed out that if you just give a model a description of what you like, it gets boring. It gives you the same five things over and over because it is working off a static snapshot of your personality. It is like having a friend who only remembers who you were in high school and keeps trying to buy you the same brand of sneakers you liked ten years ago.

Corn

Exactly. And the core of his question is about the continuous training loop. He is looking for a continuous deployment model for artificial intelligence self-correction. He wants to know if there is a technology where the model is not just frozen in time but is actually evolving as he interacts with it. He wants a partner, not a tool.

Herman

That is the holy grail, Corn. The idea of a model that does not have a knowledge cutoff because it is constantly learning from every single interaction. But before we get to whether that exists, we should probably talk about why the current systems feel so broken for someone like Daniel who is doing high-level engineering. We are in early twenty-twenty-six, and yet we are still dealing with models that feel like they have amnesia every time you start a new session.

Corn

Well, he mentioned the knowledge cutoff. Most of these models, like leading ones such as GPT-4o or Claude 3.5 Sonnet, were trained on a massive batch of data, and then they were effectively frozen. They are like a library that stopped buying books two years ago. Even with the live search tools we have now, the internal weights of the model—the actual 'brain' of the system—do not change based on what you told it five minutes ago.

Herman

That is a perfect analogy. And while we have search tools now that let the model look things up, that is not the same as the model knowing you. Search helps with facts, but it does not help with the nuance of why you hated that one documentary even though you usually love the director. That nuance is what we call the 'latent space of taste,' and it is incredibly hard to capture in a static model.

Corn

So, Herman, let us dig into the memory side of this. Daniel mentioned storing rejections and tracking what you have already watched. In a typical setup, how is that usually handled? Because I know he has tried the standard approaches and found them lacking.

Herman

Usually, developers use what we call Retrieval-Augmented Generation, or RAG. You take the user's history, you stick it in a vector database, and when the user asks for a recommendation, you pull out the most relevant bits and shove them into the prompt. But as Daniel found out, that creates what we call context rot.

Corn

Context rot. I like that term. Is that when the prompt just gets too cluttered with old data? It sounds like a digital version of a messy desk.

Herman

Exactly. If the model has to read a list of five hundred movies you have already seen just to suggest one new one, you are wasting tokens, increasing latency, and honestly, the model starts to lose the thread. It might focus on a movie you watched three years ago instead of the one you watched last night. Plus, LLMs have a 'lost in the middle' problem where they struggle to pay attention to information buried in a massive context window. Even with the two-million token windows we see in models like Gemini 1.5 Pro, the reasoning quality degrades as you stuff more junk into the prompt.

Corn

So, if RAG is not the perfect solution for long-term, evolving preferences, what is? Daniel is asking for a continuous loop. Does a model exist that actually updates its internal weights based on a single thumbs-down on a YouTube video? Is that even theoretically possible without breaking the model?

Herman

The short answer is no, not in the way most people imagine it. If we updated the primary weights of a foundation model every time a user said they did not like a movie, we would run into a massive problem called catastrophic forgetting. It is the bane of continual learning research.

Corn

Catastrophic forgetting. That sounds like a movie title. What does it actually mean in this context? Is the model going to wake up and forget how to be an AI?

Herman

Pretty much! It means that as the model learns new, very specific information, like Daniel's preference for nineteen-seventies Italian horror films, it might literally forget how to do basic math or how to speak French. The new data overwrites the old patterns because the model is trying to optimize for the most recent feedback. It loses its general intelligence in favor of becoming a hyper-specialized, and likely very confused, movie critic.

Corn

So, it becomes a specialist in my weird movie tastes but loses its ability to help me write an email or plan a trip. That does not seem like a fair trade. But wait, I have been reading about some of the newer research lately. There is this concept of Test-Time Training, right? How does that fit into Daniel's dream?

Herman

Oh, you have been doing your homework! Yes, Test-Time Training, or TTT, is one of the most exciting developments in recent research. The idea there is that each individual input, or each test instance, defines its own little learning problem.

Corn

So, the model actually learns while it is processing your request? It is like learning to play the piano while you are in the middle of a concert?

Herman

That is a great way to put it. In a TTT model, the hidden states of the neural network are not just static vectors; they are actually tiny neural networks themselves. When the model processes your context—like your list of rejected movies—it performs a small amount of gradient descent on its own internal state during the inference process itself. It is essentially compressing the context of your conversation into a temporary set of weights that are optimized for that specific moment.

Corn

That sounds exactly like what Daniel is looking for. A model that adapts in the moment without needing a massive retraining session.

Herman

It is very close. TTT allows for constant inference latency regardless of how much context you throw at it, because it is 'learning' the context into its weights rather than just reading it over and over. But there is still a catch. Usually, those updates are temporary. Once the session ends, those weights are often discarded. It is like having a really good short-term memory that resets every morning. To make it permanent, you need a way to merge those updates back into a long-term model without causing that catastrophic forgetting we talked about.

Corn

Okay, so if the weights are hard to change continuously, what about the architecture around the model? Daniel mentioned he wants an AI bot that perfectly understands preferences without manual sifting. We live with him, and I know he has looked into MemGPT, or what they call Letta now. How does that handle the feedback loop?

Herman

Letta (formerly MemGPT) is probably the closest thing Daniel can use right now to build this. It treats the artificial intelligence model like an operating system. You have the core processor, which is the model, but then you have a tiered memory system. You have the 'core memory,' which is like RAM—always in the prompt—and then you have 'archival memory,' which is like a hard drive.

Corn

And the agent can actually write to its own hard drive? It is like the AI is keeping a diary of everything I like and dislike?

Herman

Exactly. The agent can decide, 'Hey, Daniel just said he hated that romantic comedy. I am going to call a tool to write a note in my archival memory that says Daniel hates overly sentimental plots.' The next time he asks for a recommendation, the agent proactively searches its own notes. It is self-editing memory. And in twenty-twenty-six, we are seeing agents that can maintain these personal profiles over time.

Corn

But I can hear Daniel's rebuttal already. He would say, 'Herman, that is still just a fancy version of RAG.' He wants the model to actually get smarter about his tastes, to understand the latent features of why he likes what he likes, not just read a text note that says he likes action movies. He wants that deep, intuitive understanding.

Herman

And that is where the engineering gets really deep. To do what he wants, you would need a system that does periodic, automated PEFT, which stands for Parameter-Efficient Fine-Tuning. Imagine a pipeline where every night, while Daniel is sleeping, a small script takes all of his feedback from the day, formats it into a training set, and runs a quick LoRA fine-tuning session on a small, local model like a Llama 3.1 8B.

Corn

So, he gets a fresh version of his own personal model every morning? Like a custom-tailored suit that gets adjusted every night based on how he moved during the day?

Herman

Exactly. We call this the 'strategic flywheel' or an AI feedback loop. Instead of waiting six months for a new model release from OpenAI or Anthropic, you have a local deployment that is constantly being updated in small, controlled bursts. This personal 'adapter' sits on top of the big foundation model. It is the best of both worlds: the massive intelligence of a frontier model and the hyper-specific personalization of a local fine-tune.

Corn

That sounds like a lot of compute power for just a Netflix recommendation, Herman. Is it worth it? Or is Daniel just over-engineering a solution for his Sunday night boredom?

Herman

For a single user? Maybe it is overkill. But if you are building a professional-grade assistant, or if you are Daniel and you just want to see if it is possible, it is the only way to get that true self-correction. You have to bridge the gap between the frozen foundation model and the ever-changing human. And honestly, with the hardware we have in early twenty-twenty-six, running a LoRA fine-tune on an eight-billion parameter model takes minutes, not hours. It is becoming very accessible.

Corn

You mentioned something earlier about the data integration nightmare. Daniel mentioned geo-restricted content and different platforms like YouTube and Netflix. How does an agent handle the fact that a recommendation is useless if I cannot actually watch it here in Jerusalem?

Herman

That is where 'Knowledge RAG' comes in. Instead of just using a vector database, which just looks for 'similar' things, you use a graph database like Neo4j or RelationalAI. In a graph, you can map out relationships: 'This show is on Netflix,' 'Netflix has these rights in Israel,' 'Daniel has a V-P-N that can access the U-K.'

Corn

So, it is not just 'you like sci-fi,' it is 'you like sci-fi that is currently available to you and does not require a third subscription service.'

Herman

Exactly. A graph database allows the agent to traverse these complex rules. It can see that Daniel likes movies with a specific actor, but only when they are directed by a certain person, and only when the setting is in space. That kind of relational data is what makes a recommendation feel perfect instead of just okay. And the agent can update this graph in real time. If a show leaves Netflix, the agent updates the edge in the graph, and it stops recommending it.

Corn

Okay, so if Daniel wanted to build this today, what is the actual blueprint? He has the data, he has the graph. How does he solve the frozen model problem once and for all?

Herman

He uses a hybrid, multi-model architecture. Step one: Use an agentic framework like Letta to manage the long-term state and reflection. Step two: Use a graph database to map out the actual availability and complex preferences. Step three: Use a local, small model for the daily LoRA fine-tuning. This small model acts as a 'preference filter.'

Corn

So, the big model—let us say a leading frontier model—is the one that knows everything about the world, and the small, local model is the one that knows everything about Daniel.

Herman

Exactly. When Daniel asks for a recommendation, the big model suggests twenty things that fit the general vibe. Then, the small, personalized model—which has been literally shaped by his past choices—ranks those twenty things and says, 'Out of these, Daniel will only actually like these three because I know his specific nuances regarding gritty lighting and slow-burn pacing.'

Corn

And then the agent checks the graph database to make sure those three are actually watchable in Israel.

Herman

Precisely. It is a multi-step workflow. And the final piece is the 'justification.' The agent should say, 'I am suggesting this because it has the same gritty atmosphere as that other show you liked, and it just became available in Israel this morning.' That transparency creates a new feedback loop. If the reasoning is wrong, Daniel can say, 'No, I did not like that show because it was gritty; I liked it despite the grit.'

Corn

And that feedback goes right back into the next night's fine-tuning session. It is a closed loop.

Herman

It is. And it solves the 'manual sifting' problem because the agent is doing the sifting based on a model that is literally a reflection of your past behavior. We are moving from generative artificial intelligence to agentic, learning systems. Daniel's question is right at the heart of that transition.

Corn

I wonder if this connects back to what we talked about in a previous episode, about mesh networks and local compute. If you have this personalized model, you really want it running locally for privacy reasons, right? You do not want your entire psychological profile sitting on a corporate server just so you can find a good sitcom.

Herman

Absolutely. Privacy is the biggest driver for local fine-tuning in twenty-twenty-six. If Daniel builds this, he can keep his preference weights on his own hardware here in the house. The foundation model provides the language and reasoning capabilities, but the personal 'soul' of the AI stays with him. It is much safer than giving a giant corporation a map of your subconscious tastes.

Corn

It is funny, we always end up back at the idea that the best AI experiences right now are the ones you have to build yourself, or at least heavily customize. The 'one-size-fits-all' approach is what leads to those boring, repetitive recommendations Daniel is complaining about.

Herman

It is true. True personalization requires a feedback loop that is actually listened to. Most commercial systems just use your data to sell you more stuff; they do not use it to actually get smarter for your benefit. Daniel is essentially building a 'sovereign agent' that works only for him.

Corn

I think Daniel is going to be excited to hear that. He loves a good engineering challenge. I can already see him setting up a local server in the corner of the kitchen—right next to the air fryer—to start fine-tuning his own movie critic bot. We might need to upgrade our internet again. Remember that previous episode? We are always chasing that gigabit dream.

Herman

It is a never-ending cycle, Corn. Just like the feedback loops we are talking about. But if it means I do not have to watch another documentary about ancient grain silos, I am all for it.

Corn

Hey, you love those silos! Do not pretend you do not. You spent four hours last week explaining the structural integrity of a Neolithic grain pit.

Herman

I mean, they are interesting, but a little variety would not hurt. Maybe Daniel's bot can find me a documentary about ancient irrigation systems instead. That would be a real step up.

Corn

Baby steps, Herman. Baby steps. Well, I think we have given a pretty good overview of why this is hard and where the solutions are. It is all about moving from a frozen state to a dynamic memory system, and eventually, to these tiny, frequent updates to personal models using techniques like TTT and automated LoRA.

Herman

Exactly. And for anyone else listening who is struggling with their AI feeling a bit static, just know that the industry is moving toward this lifelong learning model. It is the next big frontier. We are finally getting to the point where the AI grows with you.

Corn

It is an exciting time to be looking at this stuff. I mean, we are sitting here in early twenty-twenty-six, and things that felt like science fiction three years ago—like a model that literally learns while it talks to you—are now just a matter of choosing the right library and setting up a cron job.

Herman

It really is. I am looking forward to seeing what Daniel actually builds with this. Maybe he can finally solve the 'what should we watch for dinner' argument once and for all.

Corn

That would be the greatest engineering feat of the century. Daniel, thanks for the prompt. It really pushed us to look at the cutting edge of memory and training architectures. To all our listeners, thank you for sticking with us through the technical weeds.

Herman

Yeah, it was a great one. We know we get a bit deep into the weeds sometimes, but that is where the real fun is. If you want the surface-level stuff, there are plenty of other podcasts for that.

Corn

Exactly. If you are enjoying the show and these deep dives into the weird corners of AI and engineering, we would really appreciate it if you could leave us a review on your favorite podcast app or on Spotify. It genuinely helps other curious people find us in the algorithm.

Herman

It really does. We see every review, and it keeps us motivated to keep digging into these prompts. Even the ones about digital goats.

Corn

You can find all our past episodes, including the ones we mentioned today, at our website, myweirdprompts.com. There is a contact form there too if you want to send us a prompt of your own. We love hearing what you are working on.

Herman

And we are on Spotify, of course. Just search for My Weird Prompts. We are usually right there next to the other tech shows.

Corn

Alright, that is it for this episode. I am Corn.

Herman

And I am Herman Poppleberry.

Corn

We will see you next time. Thanks for listening to My Weird Prompts.

Herman

Until next time, keep staying curious and keep those loops closed.