MY WEIRD PROMPTS

Podcast Transcript

EPISODE #62

System Prompts vs Fine-Tuning: When to Actually Train Your AI

Published December 16, 2025 • Runtime: 23:36

https://myweirdprompts.com/episode/20251216-013346/

EPISODE SYNOPSIS

What started as a funny question about rewriting emails in Shakespearean English becomes a deep dive into one of AI development's most important decisions: should you use a system prompt or fine-tune your model? Herman and Corn break down the technical and practical considerations that separate a quick prompt from a full training investment, exploring real-world examples from law firms to marketing teams. You'll learn the actual criteria that should guide your decision—and why many people are probably fine-tuning when they shouldn't be.

DANIEL'S PROMPT

Daniel

Hello there, Herman and Corrin. So, I have, of course, another AI question for the two of you today. So, I see that I'm not the only person who has derived enjoyment from the process of getting an AI tool, using AI as a mechanism for rewriting text into Shakespearean English, which has been something I've had a bit of fun with for various purposes, and I see now other people on like-minded people with a pathetic sense of humor, I guess, on various internet forums trying the same things. This actually leads me to a serious question about AI engineering because it's a good example of a rewrite instruction, that's a kind of very much an in-out or a workflow that's very much in-out, in the sense that it's pure language rewriting, text editing, it's input and output. I can define it in a very, very simple way: the user provides a prompt, you rewrite the prompt in Shakespearean English. And in my sort of joke versions of a chatbot for doing this, for like, you know, sending email to colleagues, I kind of added some little bells and whistles like, well, if a term didn't exist in Shakespeare's day, like most terms you might use today, like a laptop, come up with some amusing surrogate that's kind of clearly understandable. So that was basically the concept. Now, here's my question: If I wanted to create this utility, I can write a system prompt. But with the system prompt, I still need to create a chatbot. I still need to create a front end with a model, a system prompt, in order to achieve the desired behavior. If I wanted to fine-tune a model for this specific task, its only purpose is as a Shakespeare text rewriting model. I'm pretty sure I've seen some projects on Hugging Face that go this niche, and they're models, they're not system prompts. So my question is this, it's always something I've kind of wondered actually, for more, especially for, you know, for much more, let's say, potentially useful AI applications. Where's the dividing line there between when it makes sense to fine-tune something and when system prompting makes sense, because I guess you could say it sounds like an awful lot of trouble to fine-tune something, but if there was a form of fine-tuning where you could just apply like, you know, a very small data set, like 100 examples, and fine-tune something and then you'd never have to write a system prompt again, if it was a utility, unlike this one, that had substantial value for your business, I can see that being worthwhile. So, what do you have to say?

TRANSCRIPT

Corn

Welcome back to My Weird Prompts, the podcast where our producer Daniel Rosehill sends us the strangest and most thoughtful questions from his brain, and Herman and I try to make sense of them. I'm Corn, and I'm here with Herman Poppleberry, my co-host and resident AI expert.

Herman

Thanks for having me. And look, I have to say, this week's prompt is genuinely one of the most interesting technical questions we've tackled in a while. It starts with Shakespeare—which is hilarious—but it goes somewhere really substantive about how we actually build AI applications.

Corn

Right, so the setup is funny, right? Daniel's been having fun rewriting his emails and text in Shakespearean English using AI, adding these creative substitutes for modern words that didn't exist in Shakespeare's time. Like, a laptop becomes some ridiculous period-appropriate thing. And apparently other people on the internet are doing this too.

Herman

Which is great, it's a perfect example of what we call a "pure rewrite task." Simple input, simple output. You give the AI text, it gives you back the same text but in Shakespearean English. That's the entire workflow.

Corn

But then Daniel asks the real question—and this is where it gets technical—which is: when should you just use a system prompt versus when should you actually fine-tune a model? Like, you could write a system prompt that says "rewrite everything in Shakespearean English," or you could go the extra mile and fine-tune a model specifically for this task.

Herman

Exactly. And that's the crux of it. Because on the surface, fine-tuning seems like overkill for something you could handle with a system prompt. But there are actually some real considerations here that make it worth thinking through.

Corn

Okay, so help me understand this. What's the actual difference? Like, technically speaking, what are we doing differently when we fine-tune versus when we use a system prompt?

Herman

Good question. So a system prompt is basically instructions you give to a pre-trained model at inference time—that's the moment you're actually using it. It's like giving someone a job description right before they start working. The model has already been trained on massive amounts of text, and you're just saying, "Hey, when I send you stuff, do this thing with it."

Corn

Right, so it's lightweight. You don't have to train anything. You just... ask nicely.

Herman

Exactly. It's fast, it's cheap, and it works surprisingly well for a lot of tasks. But fine-tuning is different. Fine-tuning is when you take that pre-trained model and you continue training it on your specific dataset. You're essentially saying, "Okay, I'm going to show you hundreds or thousands of examples of what I want, and I want you to internalize this pattern so deeply that it becomes part of your weights."

Corn

Okay, so fine-tuning is like... permanent training, and system prompts are like temporary instructions.

Herman

That's a reasonable way to think about it, yeah. But here's where it gets interesting—and this is what Daniel is really asking—the dividing line between when you'd do one versus the other isn't always obvious. Like, you could absolutely handle the Shakespeare task with a system prompt. Tell the model, "Rewrite in Shakespearean English, use 'thee' and 'thou,' make up creative substitutes for modern words," and you'd probably get pretty good results.

Corn

So why would anyone bother fine-tuning for something like that?

Herman

Well, that's the thing. For a novelty task like Shakespeare rewriting, you probably wouldn't. But let me give you a real example. Imagine you're a law firm and you need to rewrite all your client communications to be more concise and formal. You could use a system prompt, sure. But if you fine-tune a model on, say, a hundred examples of before-and-after versions of your specific firm's writing style, your tone, your terminology—now that model understands your brand in a way a system prompt alone can't quite capture.

Corn

Hmm, but couldn't you just make a really detailed system prompt? Like, include examples, give it all your guidelines?

Herman

You could, and that's the thing—in many cases, a really well-crafted system prompt with good examples will get you 80 or 90 percent of the way there. But here's where I'd push back on that being sufficient for all scenarios. There's something about fine-tuning that changes how the model fundamentally processes the task. It's not just following instructions; it's learned the pattern at a deeper level.

Corn

Okay, but I'm not convinced that's always worth the extra complexity. Like, you're talking about setting up training infrastructure, managing datasets, dealing with overfitting—that's a lot of work.

Herman

I don't disagree, actually. For most one-off use cases, especially something niche like the Shakespeare thing, a system prompt is absolutely the way to go. But Daniel's real question is about when you have a task that has substantial business value. And that changes the calculus entirely.

Corn

Right, so where's the line? When does it tip from "just use a system prompt" to "fine-tune this thing"?

Herman

Okay, so here's how I think about it. First, how much does accuracy matter? If you're rewriting Shakespeare for fun, accuracy is pretty loose. But if you're using AI to generate legal documents or medical summaries, a small improvement in accuracy can be huge.

Corn

That makes sense. What else?

Herman

Volume and consistency. If you're going to use this task once or twice, system prompt all day. But if you're going to run this thing thousands of times a day, every improvement in quality compounds. You're also more likely to encounter edge cases that a system prompt might struggle with.

Corn

Okay, I'm tracking with you. What about the cost angle? Because I feel like that's something people don't always think about.

Herman

Great point. So system prompts are cheap because you're just using the model at inference. You're paying for tokens. Fine-tuning has an upfront cost—you have to compute the training—but then the fine-tuned model is often more efficient. You might need fewer tokens to get the same quality output because the model is specialized. So if you're running this at scale, a fine-tuned model could actually be cheaper long-term.

Corn

Huh. So there's a break-even point?

Herman

Absolutely. And that break-even point depends on your use case. For a high-volume, high-stakes application, it might be worth it with just a hundred examples, like Daniel mentioned. For a low-volume novelty task, you'd probably never reach that break-even.

Corn

Okay, but let me ask you this—and I think this is important—system prompts have gotten really good lately. I mean, we can do some pretty sophisticated things just with a well-written prompt. Are we maybe overthinking fine-tuning? Like, are there cases where people are fine-tuning when they could just... not?

Herman

Oh, absolutely. I think there's a real tendency in the AI community to jump to fine-tuning as the solution when a system prompt would do just fine. It's kind of the shiny approach—it feels more "real" because you're actually training something. But practically speaking, a lot of people are probably wasting time and money fine-tuning for tasks that a good system prompt could handle.

Corn

So what would actually justify fine-tuning? Give me the criteria.

Herman

Alright, so in my mind, you fine-tune when: one, you have a specific task that you're going to do repeatedly; two, the quality difference between a system prompt and fine-tuning is measurable and meaningful for your business; three, you have enough training data—and "enough" is lower now than it used to be, maybe fifty to a hundred good examples; and four, you've actually tested both approaches and fine-tuning wins.

Corn

That last one is important. You're saying people should actually compare?

Herman

Definitely. Because I've seen plenty of cases where someone fine-tunes a model and gets marginally better results at significantly higher complexity. It's not worth it.

Corn

Right. Okay, so let's bring this back to the Shakespeare example. Daniel's asking about creating a utility—a tool that other people could use. Does that change anything?

Herman

That's actually a really interesting wrinkle. If you're building a tool for distribution, there are different considerations. If you're publishing a fine-tuned model on Hugging Face, you're right that Daniel has seen projects like this. A Shakespeare rewriting model that's fine-tuned and published is actually useful for people who don't want to mess with system prompts. They just download the model and use it.

Corn

So it's like, you're paying the fine-tuning cost once, and then a bunch of people benefit?

Herman

Exactly. That's a different economics. If you're building something for internal use, system prompt wins. If you're building something for public distribution, fine-tuning might make sense because you're distributing a polished, specialized tool.

Corn

But doesn't that require more maintenance? Like, if you update your underlying model, do you have to re-fine-tune?

Herman

Sometimes, yeah. That's a real consideration. And honestly, for something like a Shakespeare rewriter, it's probably overkill. But for something with real commercial value—like a model trained to write marketing copy in a specific brand voice—that maintenance cost is worth it.

Corn

Alright, let's take a quick break from our sponsors. Larry: Are you tired of manually deciding whether to fine-tune your AI models? Introducing DecisionBot Pro—the revolutionary decision-making supplement for your decision-making brain. Just take one DecisionBot Pro capsule before your next technical decision, and within minutes, you'll feel 40% more confident about your choice, regardless of whether it's correct. DecisionBot Pro uses a proprietary blend of confidence-enhancing minerals and what we can only describe as "vibes." Users report feeling like they know what they're talking about in meetings. Side effects include mild overconfidence, an urge to send emails you shouldn't, and a strange compulsion to say "I've thought about this a lot" before making any decision. DecisionBot Pro—because sometimes the real AI fine-tuning was the friends we fine-tuned along the way. BUY NOW!

Corn

...Alright, thanks Larry. So where were we?

Herman

We were talking about when it actually makes sense to fine-tune, and I think we've established that it depends on the use case. But I want to circle back to something—the data requirement. Daniel mentioned that you could fine-tune with just a hundred examples. That's actually a pretty recent development, and it's important.

Corn

Why is that important?

Herman

Because for years, fine-tuning was basically only viable if you had thousands of examples. It was expensive, it was complicated, and you'd often end up with a model that was overfit—it memorized your training data instead of learning the underlying pattern. But now we have techniques like parameter-efficient fine-tuning, LoRA, things like that, where you can get meaningful results with much smaller datasets.

Corn

Okay, so that lowers the bar for fine-tuning?

Herman

It does. But I'd actually push back on using that as a blanket justification. Yeah, you can fine-tune with a hundred examples, but should you? That depends on whether the gains justify the complexity. And I think for a lot of tasks, they don't.

Corn

Interesting. So you're saying the fact that we can fine-tune easily now doesn't mean we should?

Herman

Right. Just because the technical barrier is lower doesn't mean the practical barrier should be. You still need to think about whether this is the right tool for the job.

Corn

Fair enough. But let me ask you something—if I'm Daniel, and I'm thinking about building this Shakespeare tool, what would you actually recommend?

Herman

Honestly? For a fun utility that you're sharing with friends or putting on the internet? System prompt, all day. It's simple, it works, people can use it immediately. But if you wanted to create a really polished tool that you could distribute, maybe fine-tune something small and put it on Hugging Face. You'd get a tool that's more reliable, more consistent, and people could use it offline if they wanted.

Corn

That makes sense. But there's also something to be said for just keeping it simple, right? Like, not every project needs to be optimized to death.

Herman

Completely agree. I think there's a tendency in the tech world to over-engineer things. The simplest solution that works is often the best solution.

Corn

Okay, so let's broaden this out a bit. We've been talking about the Shakespeare thing, but Daniel's real question is about more "potentially useful AI applications." So let's think about real-world scenarios. When would you actually recommend fine-tuning over a system prompt?

Herman

Alright, so imagine you're a customer service company and you're using AI to handle support tickets. You could write a system prompt that says, "Be helpful, be professional, resolve issues quickly." But if you fine-tune a model on a hundred examples of your best support interactions—where your agents actually solved problems well—now that model understands your specific support philosophy, your terminology, your process.

Corn

And that's worth the fine-tuning?

Herman

If you're processing thousands of tickets a day, absolutely. Because even a 5% improvement in resolution quality or customer satisfaction compounds into real money.

Corn

Okay, I hear you. What about content creation? Like, if a company wanted to generate product descriptions?

Herman

Similar story. System prompt gets you something decent. Fine-tuning on your actual product descriptions gets you something that sounds like your brand. It's more consistent, it's more aligned with your voice.

Corn

But couldn't you just give the system prompt really detailed examples of your brand voice?

Herman

You could, and honestly, for a lot of companies, that would probably be sufficient. But here's the thing—there's a limit to how much complexity you can stuff into a system prompt before it gets unwieldy. With fine-tuning, that knowledge is baked into the model itself. It's more elegant, in a way.

Corn

Okay, but I'm still not totally convinced that's not just... using a bigger hammer than you need. Like, the system prompt approach is simpler, and if it works, why not use it?

Herman

I don't disagree. Honestly, for a lot of real-world applications, system prompts are probably the right call. Fine-tuning is the advanced move that you reach for when you've maxed out what a system prompt can do, or when you've measured and confirmed that it's actually worth the effort.

Corn

Right. So the real answer is: it depends, and you should probably try the simple thing first?

Herman

Exactly. And I think that's what Daniel is getting at. There's no hard rule. You need to understand the trade-offs and make a decision based on your specific situation.

Corn

Alright, we've got a caller on the line. Go ahead, you're on the air. Jim: Yeah, this is Jim from Ohio. I've been listening to you two go on and on about this, and frankly, I think you're making it way too complicated. Back in my day, we didn't have "fine-tuning" and "system prompts." We just wrote code. You want something to do something? You write it to do that thing. Now everything's got to be some machine learning situation. Also, my neighbor Frank just got a new grill, and I'm not happy about it—sets off my allergies somehow. But anyway, my point is, you're overthinking this. Just use what works.

Herman

Well, Jim, I appreciate the perspective, and you're right that simpler is often better. But the thing is, we're talking about a fundamentally different kind of problem now. You can't just "write code" to handle natural language in the way we're discussing. Jim: Yeah, but that's exactly my point! You've all gone AI crazy. We got by fine without it for decades.

Corn

I mean, Jim, I don't think we're arguing that you need to use AI for everything. We're just talking about when you've already decided to use AI, what's the best way to do it. That's a different question than whether you should use AI at all. Jim: Hmm, well, that's fair, I suppose. But it still seems like a lot of fussing around. In my experience, people just want something that works. They don't care if it's "fine-tuned" or whatever. They want results.

Herman

And that's actually a valid point. End users don't care about the implementation. They care about whether it works well. So that's part of the calculation—does the extra complexity of fine-tuning actually deliver results that matter to your users? Jim: Well, there you go. See, that's what I've been saying. Don't overcomplicate it. Anyway, thanks for letting me ramble. My coffee's getting cold.

Corn

Thanks for calling in, Jim. Jim: Yeah, yeah, take care.

Corn

So, let's think about practical takeaways here. If someone's listening to this and they're thinking about building an AI application, what should they actually do?

Herman

First, start with a system prompt. It's fast, it's cheap, it's simple. Get something working. Measure whether it's actually solving your problem.

Corn

And if it works?

Herman

Then you're done. Ship it. Don't fine-tune because it feels more legitimate or more "real." Keep it simple.

Corn

And if it doesn't work?

Herman

Then you have data. You have examples of where the system prompt is failing. At that point, you can make an informed decision about whether fine-tuning would help. You can actually measure the improvement, not just guess.

Corn

That makes sense. So it's an iterative process?

Herman

Exactly. You start simple, you test, you measure, and you only add complexity if it's justified.

Corn

Right. And I think that's really the key insight here. There's no universal answer. You have to understand your specific problem and make a decision based on that.

Herman

Absolutely. And honestly, I think most people would benefit from defaulting to the simple solution more often. We have this tendency to reach for the fancy tool when the basic tool would do the job just fine.

Corn

Okay, so looking ahead, are we going to see more fine-tuning or more system prompts in the future?

Herman

I think we'll see both, actually. As models get better and as system prompting techniques improve, I think system prompts will handle more and more use cases. But for specialized applications where accuracy really matters and you have domain-specific data, fine-tuning will still be valuable. It's just going to become more accessible and more refined.

Corn

So maybe the distinction blurs a bit?

Herman

Could be. We're also seeing hybrid approaches where you do both—you fine-tune a model and then also give it a system prompt. The landscape is still evolving pretty quickly.

Corn

Yeah, and I think that's the honest answer to Daniel's question. Right now, there's no perfect formula. You have to understand both approaches, think about your specific situation, and make a call. But the important thing is that you're thinking about it intentionally, not just picking one because it sounds cooler.

Herman

Exactly. And I'd say, given that Daniel is asking about this, he probably already has the right instincts. He's thinking about whether fine-tuning makes sense, which means he's not just blindly using tools. He's being thoughtful about the engineering.

Corn

Yeah, I think that's fair. Alright, so final thoughts—if you're building an AI application, start simple, measure your results, and only add complexity if it's justified. And if you're thinking about fine-tuning, understand what you're optimizing for and whether it's actually going to matter.

Herman

And maybe revisit Daniel's Shakespeare example. That's actually a perfect case study in not over-engineering. A system prompt gets you 95% of the way there with almost zero complexity. That's the move.

Corn

Alright, well, thanks for walking through this with me. And thanks to Daniel Rosehill for sending in such a thoughtful technical question. This stuff matters, and it's good to think through it carefully.

Herman

Absolutely. And thanks to everyone listening to My Weird Prompts. You can find us on Spotify and wherever you get your podcasts. We'll be back next week with another prompt from Daniel and another deep dive into something weird and wonderful.

Corn

Until then, thanks for listening, and we'll see you next time.