Episode Book

EPISODE #5604

Fixing Typos Without Rewriting the Prompt

PUBLISHED	27 September 2026 at 11:16 UTC
DURATION	20:33
LISTEN	myweirdprompts.com/episode/prompt-typo-correction-pipeline/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:38 UTC

SYNOPSIS

Daniel dictates his prompts, catches typos by hand, and keeps running into the same wall: fixing a misspelling is not the same job as understanding what he meant. This episode pulls apart the pipeline into its three real tasks — disfluency removal, mechanical typo correction, and semantic inference — and asks which of them a language model should be allowed to touch. Along the way: why an "um" is a token like any other, why a confidence threshold is a certainty parameter you just reinvented, and why a small task-specific corrector drifts less than a large general model. If you've ever handed a "just fix the typos" prompt to a model and gotten back something smoother than what you wrote, this one's for you.

TRANSCRIPT



Corn

Here's what Daniel wrote in this week. He dictates his prompts into his phone, then goes through and catches the typos by hand. Mostly it works. Every so often one slips through, and Parakeet, which he quite rightly calls excellent, transcribes his disfluencies with total fidelity. So he's got two problems sitting in the same sentence, and he's noticed they're not the same problem.



Herman

Which is the interesting part.



Corn

He says he usually leaves the ums and uhs in, on the theory that nobody meant them as part of the prompt. But he wonders whether there's a technical case for stripping them, because they're garbage tokens diluting the input vector space.



Herman

That's the first question.



Corn

Then he thought about bolting a spell-check layer onto the front. Except a spell-check layer needs a confidence threshold. Fix the obvious misspelling, leave the ambiguous one alone. And he caught himself mid-thought, because a confidence threshold is a certainty parameter, and he'd just reinvented one by accident.



Herman

He does that a lot.



Corn

Then the real turn. He decides this isn't a spell-check problem at all, it's a reasoning problem. The ideal instruction is something like: if it's clear from context that this was a typo, especially a cognate, take the replacement. If not, leave it. Which means semantic understanding. Which means a model.



Herman

And that's where the nerves start.



Corn

That's where the nerves start. Because all of it could go through one language pass, disfluencies and typos together. But it's a bounded correction task, and probabilistic generation models make him nervous that they'll get generative instead of remediating. He's seen the new class of post-processing models, S1 Mini and its relatives, built for exactly this. His objection is that they ship with the authors' cleanup rules rather than the ones he wants to control for. So: small instruction model with a system prompt? Long-standing spell-check package for the mechanical stuff with AI reserved for semantic inference? Some combination? That's the ask.



Herman

Good. Let's do it properly, because there are three different jobs hiding in that one pipeline.



Corn

Name them.



Herman

Disfluency removal. Typo and capitalization correction. And semantic inference correction, the context-dependent kind. Those look like one task when you write them in a list. They are not one task. They need different levels of reasoning and they fail in different ways, and the single biggest mistake people make is running all three through one generative pass and calling it a sanitization layer.



Corn

So the architecture question is really a question about which of those three you're allowed to hand to a model.



Herman

Right. And the answer is not the same for all three. Start with the disfluencies, because Daniel's instinct there is half right and the half that's wrong is the interesting half.



Corn

He thinks they're garbage tokens.



Herman

They are tokens. That part is true and I think people skip past it. An um is not filtered out before it reaches the model. It gets tokenized, it gets an embedding, it takes up a position in the sequence, and it participates in attention like everything else. There's no janitor at the door throwing out the filler.



Corn

So it does occupy space in the vector.



Herman

It occupies space. Whether it dilutes anything is a different claim. Modern models are trained on an enormous amount of transcribed speech, and transcribed speech is full of this. Interviews, lectures, podcasts, court transcripts. The model has seen millions of ums in context and learned, roughly, to weight them low. Mild disfluency is close to a non-issue for instruction following. You can hand a model a paragraph with four uhs in it and get the same answer you'd get from the cleaned version.



Corn

Then where's the harm?



Herman

It shows up at the margins. When you push filler density up, the attention weights start to shift, and tasks that need precise instruction following get worse. Long prompts, many constraints, several things to track at once. The filler isn't poisoning the meaning, it's spending attention budget that the constraints needed. So the honest answer to Daniel's first question is: leaving them in is fine most of the time, and the case for removing them gets stronger as the prompt gets longer and more specific.



Corn

Which is exactly the shape of the prompts he sends us.



Herman

So for his use case, strip them. But I want to flag something before we move on, because it comes back around at the end of the episode and I don't want it to arrive as a surprise.



Corn

Go on.



Herman

An um is not always noise. Sometimes it's a person working out what they mean in real time. If you strip every one of them you're removing a record of hesitation, and hesitation is information. Hold that thought.



Corn

Holding it. Now the typo layer, because that's where he caught himself.



Herman

That's the confidence threshold, and his instinct is correct, and the reason it's correct is that the alternative is worse. A corrector with no threshold will fix things that aren't broken. So you want a system that can abstain. The literature calls it selective prediction, and the point of it is that a model should be allowed to say nothing rather than guess.



Corn

That sounds obviously right. What's the catch?



Herman

Calibration. A model's confidence score is not the same thing as its accuracy. It'll tell you it's ninety-five percent sure about a correction it gets wrong, and it'll be uncertain about one it gets right. So you set a threshold, and the threshold does two jobs badly. Too low, and you're accepting confident errors, which is the worst outcome because the user never sees them. Too high, and the system abstains on everything, and you've built an expensive no-op.



Corn

And there's no clean number.



Herman

There's no clean number. You tune it against your own data and you accept that you're choosing which failure you'd rather have. For a prompt sanitizer, I'd bias hard toward abstaining. A typo that survives is a small annoyance. A confident wrong correction is a silent change to what you asked for.



Corn

Which is the thing he's actually scared of.



Herman

It's the thing he's actually scared of, and he's right to be. Here's the cleanest example I can give you. Daniel writes a prompt that says, write a poem about the see. S-E-E. A spell-checker looks at that and sees a real word. Every letter valid, in the dictionary, nothing to flag. An LLM looks at it and sees a word that doesn't fit the sentence, and infers sea. And nine times out of ten that's the right call.



Corn

And the tenth time?



Herman

The tenth time he meant see as in observe, and the model just rewrote his instruction. That's the whole problem in one sentence. The correction requires understanding what he meant, and understanding what he meant is not something a dictionary can do.



Corn

So the spell-checker's failure is that it can't see the error. The model's failure is that it can see an error that isn't there.



Herman

It gets worse across languages, which matters for him specifically. Cognates. Actual in English means real or existing. Aktuell in German means current, happening now. A model doing semantic cleanup on mixed-language dictation will happily correct a word that was right into a word that's wrong, and it'll do it confidently, because from where it's standing the sentence reads better afterward.



Corn

And he dictates. So half his errors aren't spelling at all, they're the ASR mishearing him.



Herman

Which is a third category and it deserves its own name. A typo is his fingers. A mishearing is the model's ears. They look identical in the output and they need different fixes. A misspelling you can catch with edit distance. A mishearing you can only catch with context, because the word that came out is a real word that just isn't the one he said.



Corn

So we've got three failure classes and one of them is a reasoning problem.



Herman

Two of them are. Semantic correction and mishearing repair both need the model to understand the sentence. The mechanical layer can't touch either. And that's why his instinct that this is an AI problem is right, and also why his nerves are right, because the moment you hand a correction task to a generative model you've handed it a task with no hard constraint on preserving everything else.



Corn

Explain that. Why does it drift?



Herman

Because of what you asked it to do. You said fix the typos. The model doesn't have a switch that says fix typos and only typos. It has a distribution over what a helpful response looks like, and a cleaned-up version of your text is in that distribution, and so is a slightly rephrased version, and so is a tightened version, and so is a summary if the input was long enough to look like it wanted summarizing. Nothing in the request forbids any of that. So sometimes you get it.



Corn

And you don't notice, because the output reads fine.



Herman

The output reads better, that's the trap. It's smoother than what he wrote. It's just not what he wrote.



Corn

Is that a size problem? Bigger model, more restraint?



Herman

It's the opposite, and this is the counterintuitive bit. The models that are least prone to over-generation on correction tasks are the small ones that were fine-tuned specifically for correction. A large general-purpose model is more capable, which means it has more ways to be helpful, which means more ways to drift. Narrow the task and narrow the model and you get less drift, not more.



Corn

So the small instruction model isn't the compromise option. It's the correct one.



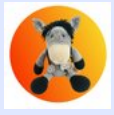
Herman

For this task, yes. And that's a real finding, not a hunch. Task-specific correctors over-generate less than general models doing correction as a side quest.



Corn

Alright. So that's the why. What does the stack actually look like?



Herman

Three stages, and the order matters. Stage one is deterministic. No model, no probabilities, no judgment. This is where you strip the clear fillers, restore capitalization, fix the obvious misspellings, collapse the doubled words. A spell-check library has been doing most of this for decades and it does it the same way every time, which is the entire virtue. Hunspell, LanguageTool, whatever you like. It handles the large majority of real corrections with zero generative risk, because there's nothing generative in it.



Corn

And when it's wrong?



Herman

It's wrong in boring ways. It flags a name, it flags a technical term, it suggests a word you didn't want. You see it immediately, it's the same every time, and you add an exception. That's a failure you can live with.



Corn

Stage two.



Herman

Stage two is the small instruction-tuned model, and only for what stage one can't reach. The homophones, the cognates, the mishearings, the word that's real but wrong. And the system prompt is doing real work here, not decoration. You constrain it explicitly. Correct only clear errors. Do not rephrase. Do not add content. Do not remove content. If uncertain, return the input unchanged. And then you show it examples, because few-shot beats instruction every time on this kind of task. Here's an input, here's the correct output, here's an input where the right answer was to change nothing.



Corn

That last category is the one people skip.



Herman

It's the most important one. If every example you show the model involves a correction, you've taught it that corrections are what you want. You need examples where the correct behavior is to leave it alone, or it'll find something to fix in every sentence.



Corn

And the confidence output.



Herman

Have it emit a confidence token alongside the correction, and gate on it in the pipeline. Below the threshold, discard the model's output and pass the original through untouched. That's the abstain path, and it should be a normal outcome, not an error state.



Corn

Stage three.



Herman

Stage three is the one I'd argue is non-negotiable and the one most homegrown pipelines don't have. You compare the input and the output. Edit distance, a diff, whatever's cheap. If the model changed more than the correction budget allows, you reject the whole thing and fall back to the original.



Corn

Because the failure you can't see is the one that hurts.



Herman

Because a model that quietly rewrote a paragraph will never tell you it did. The diff will. It's a few lines of code and it catches exactly the failure mode Daniel is worried about, which is the model getting generative instead of remediating. You don't need to detect what it changed. You just need to notice that it changed too much and throw the whole pass away.



Corn

That's the guardrail.



Herman

And it's cheap, which is why I don't understand why it isn't standard.



Corn

Now the off-the-shelf option, because he raised it and I think it deserves a fair hearing.



Herman

S1 Mini and the models in that family are good at this. Fast, consistent, built for exactly the transcript-polishing job, and if your notion of clean matches theirs you're done in an afternoon. The problem is that their notion of clean is baked in. They were trained on a particular set of cleanup rules and they will apply those rules to your text whether or not you wanted them.



Corn

Give me the concrete version.



Herman

Say the model was trained to remove all filler and restore full punctuation. That's a reasonable definition of clean. Now say you're dictating something where you want the hesitation preserved, or where you're deliberately writing in fragments. The model doesn't know you have a reason. It cleans it anyway, because cleaning is what it does. You can't tell it to leave the fragments alone, because that instruction isn't in the weights.



Corn

So the tradeoff is speed and consistency against control.



Herman

Against control, and control is the whole point of the exercise. He's not trying to make transcripts pretty. He's trying to make sure what reaches the model downstream is what he meant. Every cleanup rule that isn't his is a small loss of that.



Corn

So the recommendation, if you had to give one.



Herman

Hybrid, and I'd be specific about the split. Rule-based layer first for everything deterministic, because it's free, it's fast, and it can't hallucinate. Small instruction-tuned model second, gated on confidence, constrained by a system prompt with negative examples. Validation pass third, edit distance, reject on overreach. And no large general-purpose model anywhere in the loop.



Corn

Not even as a fallback for the hard cases?



Herman

That's the one place I'd consider it, and I'd still gate it. If stage two abstains and the edit distance is small, a bigger model might resolve it. But you're back to the drift risk, so it needs the same validation pass, and honestly at that point you're spending a lot of machinery on the residual cases. For a personal pipeline, I'd leave them. A typo that survives to the model is a smaller problem than a correction you didn't authorize.



Corn

And the disfluency question, one more time, because we parked something.



Herman

We did. Strip the clear fillers. Leave the false starts. A false start is usually a person correcting themselves mid-sentence, and if you delete it you've deleted the correction and kept the thing they were correcting. That's the one case where aggressive cleanup makes the text worse than the raw input.



Hilbert

The clinic ran a three-um rule.



Corn

Sorry, go on.



Hilbert

Medical transcription. Doctors dictating notes, two years of it, before the machines got good at it. We typed what they said and cleaned it as we went. House style was never change a word unless you were certain it was wrong, and even then you flagged it for the physician to sign off. Certain. Not probably.



Herman

Which is a higher bar than most pipelines set.



Hilbert

It was higher than we liked. But the three-um rule was the interesting one. If a doctor said um more than three times in a single sentence, you left every one of them in. That was the instruction. You didn't clean it.



Corn

Why?



Hilbert

Because three ums in one sentence meant he didn't know what he was writing. The note would go back to him with the hesitations intact so he could see where he'd stalled. The ums weren't noise. They were the part of the note that told you which part not to trust.



Herman

That's a confidence signal, encoded in the filler.



Hilbert

It was a flag. We had a woman on the desk, Ruth, who'd been doing it fifteen years, and she could read a dictation and tell you which line the doctor was going to change before he changed it. She was right more often than the software they eventually bought to replace us.



Corn

And the software stripped the ums.



Hilbert

The software stripped everything. That was the selling point.



Corn

So what happened to the notes?



Hilbert

They got cleaner. I don't know that they got more accurate. Nobody ever measured that, as far as I know. The physician signed off either way.



Herman

The thing I keep circling back to is that the rule wasn't about the ums at all. It was about preserving the record of uncertainty. You can build that into a pipeline, but only if you decide in advance that uncertainty is something you want to keep.



Corn

And most cleanup rules are written by people who've decided the opposite.



Hilbert

Most cleanup rules are written by people who want the text to look finished.



Corn

Before we close this out, I want to put the practical version on the table, because Daniel asked for a recommendation and we should actually give him one.



Herman

Start with the deterministic layer. Spell-check library, capitalization, obvious misspellings, clear fillers, and a list of exceptions you maintain by hand. That's most of the work and none of the risk. Then the small instruction-tuned model, gated on a confidence threshold you've tuned to abstain aggressively, with a system prompt that names the forbidden behaviors and few-shot examples that include cases where the right answer is to change nothing. Then the validation pass, edit distance against the input, reject anything that moved too far.



Corn

And the threshold, if he has to pick a starting point.



Herman

Bias it toward leaving things alone. A surviving typo costs him a re-read. A silent wrong correction costs him the whole prompt, and he won't find out until the output is wrong for reasons he can't trace.



Corn

One thing I'd add, and it's the thing Hilbert just handed us. If disfluencies can carry a record of where the speaker stalled, then the cleanup layer is making an editorial decision every time it strips one, and it's making it silently. That's worth knowing before you write



Corn (continued)

the rule.



Herman

Which changes the shape of the question a bit. It's not just what should we clean. It's what are we choosing to forget.



Corn

If you take one thing from this, take the ordering. Deterministic first, model second, validation third. The rule-based layer isn't the boring part you skip to get to the AI. It's the guardrail that makes the AI safe to use.



Herman

The validation pass is what makes it bounded. Without it you don't have a correction pipeline, you have a model with an opinion about your writing.



Corn

The open question, then. As these post-processing models get better, the line between cleaning text and rewriting it is going to keep blurring, and the systems that hold that line are the ones that can tell the difference. That's a reasoning problem, not a text-processing problem, and it's the part that doesn't get easier just because the models get bigger.



Herman

Hilbert's three-um rule is the version of that I'll remember. Somebody decided that uncertainty was worth keeping, and wrote it down, and it worked.



Corn

Thanks to Hilbert Flumingtop, our producer. This has been My Weird Prompts. If you're building anything like this, the show notes are at my weird prompts dot com, and you can email us at show at my weird prompts dot com. We'll be back soon.



Herman

See you tomorrow.