Episode Book

EPISODE #6159

When a Speech Model Can't Decide How to Write Dollars

PUBLISHED	20 October 2026 at 13:48 UTC
DURATION	18:33
LISTEN	myweirdprompts.com/episode/speech-model-currency-convention-clash-28065/
HOSTS	Corn, Herman
SCRIPT MODEL	anthropic/claude-sonnet-5.5
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.10
BOOK GENERATED	21 October 2026 at 07:50 UTC

SYNOPSIS

Daniel is fine-tuning Parakeet v2, a 600-million-parameter speech model, on his own voice — English and Hebrew mixed in the same sentence. His training data is built as small, atomic datasets, one per phenomenon: currency, brand names, acronyms, self-corrections, code-switching. But when one module writes "$20" and another writes "twenty dollars," does the model pick a winner, split the difference, or invent a hybrid nobody labelled? This episode digs into how convention clashes hide inside fine-tuning pipelines, why naive string-diffing cries wolf, why confident-learning tools are the wrong instrument for formatting disagreements, and what a single canonical normaliser can catch in seconds that a model would never flag.

TRANSCRIPT



Corn

What does a speech model do when one half of its training data says "write the dollar sign" and the other half says "write the word dollars"? Does it pick a winner, split the difference, or just get confused in public? That's the thing Daniel ran into while reading sentences aloud on his phone for his Parakeet fine-tune.



Herman

Parakeet, the six hundred million parameter one, version two. He's adapting it to his own voice, mostly for English and Hebrew mixed in the same sentence. And he's built the training data the way a tidy engineer would. One small dataset per phenomenon. Currency, brand names, acronyms, self-corrections, Irish English, code-switching. Each one is its own subset on Hugging Face, so a run can take any one or leave it.



Corn

Claude Haiku writes the sentences, he reads them out, and his picture of the whole thing is a blender. Each dataset is a piece of fruit, and a recipe is what you throw in for a particular adaptation.



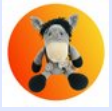
Herman

Then in one set he spots a lowercase "dollars" sitting there, while the currency set insists on the symbol with the number. Twenty dollars becomes a dollar sign and a two and a zero. So his worry is that two modules are pulling in opposite directions, and nobody notices until the model gets weird.



Corn

Before I worry on his behalf, I'd like to know whether those two actually disagree.



Herman

That's where I'd start too, and it's a good example of how these checks go wrong. His convention, as written down, is that an amount gets the symbol in front. Dollar sign, twenty. But a currency named without an amount stays a word. "The dollar fell." "Some shekels." So a lowercase "dollars" is perfectly correct if no number is standing next to it.



Corn

So the contradiction might be a sentence about, I don't know, dollars in general.



Herman

Might be. It becomes a real conflict only if the sentence says "twenty dollars" and the target says "twenty dollars" in words, while the currency set says the symbol for the same spoken input. A conflict is defined by the same input pattern mapping to different targets. Not by two strings looking different on a screen.



Corn

Which means a script that diffs the two datasets for the word "dollars" would flag that sentence and cry wolf.



Herman

Constantly. Naive string comparison over-reports massively, because any sensible style guide has context rules in it. Digits for amounts, words for the bare noun. I haven't seen the exact sentence, so I can't tell Daniel which it is. But step one is always: pull the sentence, look for the number beside it.



Corn

And if the number is there?



Herman

Then it's a real clash, and here I should be straight about the limits. Nobody has put a number on how a model resolves that exact case in speech recognition. What I can reason out is this. The training loss pushes probability toward each target in proportion to how often it shows up for that kind of audio. So if the symbol form appears in most of the examples, it wins most of the time. But the decoder emits one piece at a time, and each step takes the most likely next piece. So you can get a splice. A dollar sign, and then the word "twenty" after it.



Corn

A hybrid nobody wrote.



Herman

Nobody wrote it, nobody labelled it, and it will look like a bug in the model. The other thing is that small cues can tip it. If the symbol form always appeared in clipped, newsy sentences and the word form in chatty ones, the model may learn the register as the switch. A preprint from this past July puts it nicely. Models trained on a mix of annotation styles treat the style as an uncontrolled latent variable. Something the model has to guess, and you never told it to.



Corn

Their fix being?



Herman

Special tokens at the start of the output that say which style you want, trained on paired examples of the same speech written both ways. Which is a big step up from "hope the majority wins." I only read the abstract, mind you.



Corn

Hope the majority wins sounds like most of my life decisions.



Herman

Hm. Let me give you a concrete case from Daniel's own repo, because it's sharper than the dollars. On October twentieth he changed the convention for shekels. Money in shekels now gets written as NIS after the number. Five recorded takes on Hugging Face were rewritten, and the old text is kept in a before-normalisation field.



Corn

Fine, a sensible change.



Herman

The published first version of the model was trained on October nineteenth. The day before.



Corn

So version one learned the old way.



Herman

By the dates, presumably yes. I want to be clear that's inferred from the calendar. Nobody has run the model and looked at what it prints for a shekel amount. But the repo's own rule says a normalisation pass that rewrites transcripts is a reason to retrain, and this one landed after the run.



Corn

So the model that's out there has opinions about shekels from a day before the memo.



Herman

Which is also a quiet reminder that the tokenizer matters. It has the dollar, euro and pound symbols, and no shekel sign. The model can't emit one even if you wanted it to. So "NIS" isn't just a style call, it's what the vocabulary can actually say.



Corn

I'd have thought the symbol was the whole point of a currency set.



Herman

For three of the currencies, sure. For the fourth, which is the one he cares about most, it's letters.



Corn

Which brings us to Daniel's real question. Atomic datasets.



Herman

And I think the interesting discovery is that his own setup isn't as atomic as the pitch.



Corn

Go on, I can feel a donkey about to be thrilled.



Herman

I am a little thrilled. So the app has task tags, and those really are opt-in. A currency run takes currency, a hard-ASR run takes hard-ASR, and a tagged task stays out unless you name it. But the bank categories behave differently. Code-switching is always in training. And religious and yiddish rows sit in that share, so they train whether or not you name their tag.



Corn

So leaving one fruit out of the blender works for some fruit and not for the rest.



Herman

There are two kinds of ingredient. The tagged ones you can drop. The bank ones are in the base, and Yiddish and religious material overlap code-switching by design, which is reasonable. It just means "atomic" is a spectrum rather than a property.



Corn

Is that a problem, or is it just a fact to write down?



Herman

Mostly a fact. It becomes a problem when a convention in a bank category disagrees with a convention somewhere else, because you can't ablate it away. Which leads to the Harvard sentences.



Corn

The ones with the stale smell of a sixties phonetics lab.



Herman

Those. They're in the hard-ASR set, and one of them keeps "fifty bonds" spelled out as words. Fine in the code-switching run, because hard-ASR is out of that run by default. But Daniel plans a second fine-tune, English only, no Hebrew, and hard-ASR is meant to go into that one. And that run also has the rule that numbers of ten and above are digits.



Corn

So "fifty bonds" would be a live collision, but only in a run that hasn't happened yet.



Herman

Which is the nasty kind. A conflict that depends on which fruit you picked. Looking at one dataset in isolation tells you nothing. You only see it in the recipe.



Corn

So what do you do about the ones you can't see by eye? That's what he's actually asking, the non-obvious ones.



Herman

I'd put it to you the other way round first. If you were building this, where would you expect the quiet contradictions to hide?



Corn

In whoever writes the sentences. Haiku has its own taste, and every dataset gets generated by a separate prompt. Each prompt carries its own idea of how a number is written.



Herman

That's the soft spot. If the style rules live in ten generator prompts, you've got ten slightly different style guides. One canonical normaliser, one style guide, as the single source of truth. Then each module's generator prompt is derived from it, rather than hand-written alongside it.



Corn

So the dollars wouldn't have a chance to drift, because nobody gets to type the rule twice.



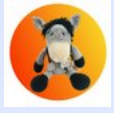
Herman

And the normaliser also doubles as a detector. Run it over every module's targets. Anything it would change is a place where that module disagrees with the canon. That's cheap, it takes seconds, and it needs no model at all.



Corn

Daniel's own dry-run of that normaliser, if I remember the repo, wanted to change a hundred and thirty-two casings.



Herman

Some of which were wrong, the notes say. So you can't just apply it blindly. It's a flag-raiser, and a human decides whether the module or the normaliser is the one in error.



Corn

Which is annoying, because it means the detector has bugs too.



Herman

Everything has bugs. It's the same input being checked by two things that should agree. Where they don't, you look.



Corn

Hm. He's already caught one of these on his own, hasn't he? The replay data.



Herman

The LibriSpeech swap. When you fine-tune on a narrow voice, you mix some of the original model's domain back into each batch, so it doesn't forget everything it knew. One study mixed in ten percent and the old-domain error rate on a small Whisper model dropped from about twenty-four and a half to about eighteen. Only ten and twenty percent were tested, so don't treat ten as magic.



Corn

And the contradiction?



Herman

The repo notes that the transcripts for that replay set, as hosted on the Hub, come out uppercase with no punctuation. I haven't looked at them myself. But the model card for version two says its training transcripts keep punctuation and capitalisation. So replaying uppercase, unpunctuated text would be a module telling the model to unlearn exactly what the base model was trained on.



Corn

Replaying data meant to protect the model, and it undoes the capitalisation instead.



Herman

Which he spotted and swapped. And note what kind of problem it is. Nothing's wrong with the words. The words are correct. It's the convention that's wrong.



Corn

That feels important, and I can't tell why yet.



Herman

Because it decides which tools work. There's a family of methods for noisy labels. Confident learning is the best known. You train a model, look at its out-of-sample predicted probabilities, estimate how given labels and true labels relate, and prune the likely errors. There's a package for it, cleanlab. And there's dataset cartography, which tracks how confident the model is about each example across epochs in one run. The hard-to-learn ones often turn out to be mislabelled.



Corn

So run those over the blend.



Herman

They find wrong words. A transcript that says "cat" where the speaker said "cap." They're much worse at finding wrong conventions, because a convention is consistently applied inside its own module. The model learns it perfectly. The label is not noisy, it's just a different dialect. So I'd call them the wrong tool for this class. One follow-up on cartography even found selected subsets didn't help generalisation on a question-answering set.



Corn

So for formatting clashes you want the boring tool, and for typos you want the clever one.



Herman

Boring first. Corn, what would you do next, if the normaliser comes back clean?



Corn

Train on one module and test on another. If the model trained only on the currency set writes the symbol, and then I feed it the sentences from the other tasks and it keeps writing symbols where the labels say words, the disagreement shows up as error. It's not a proof. It's a smoke alarm.



Herman

That's a good one, and it's cheap because you've got the data already. You'd see the specific rows where the two modules disagree, without having to guess in advance which ones to read. And then there's the leave-one-module-out ablation.



Corn

Make the blender without the banana and see if it tastes different.



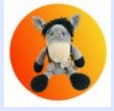
Herman

Fair, though I'll note, as a suggestion rather than a published finding, that it only works for the tagged tasks. The bank categories stay in. Anyway, train with and without a module, and compare each module's own eval set. If the currency score goes up when you drop the acronym module, they're fighting.



Corn

Each module gets its own eval set. Fine, but the number I'd worry about is the one that stays flat for no reason, because your reference changed and not your model.



Herman

Oh, that one's a real trap. Whisper's work taught this. They run an English normaliser over both the reference and the model's output before they compute word error rate. It expands contractions and currency symbols so the score isn't about spelling. If you



Herman (continued)

change a convention in your references and don't normalise both sides the same way, a convention change looks like the model got worse.



Corn

So the model didn't regress. The goalposts moved nine inches to the left.



Herman

Same function on both sides, always. Then a regression is a regression.



Corn

There's also the opposite worry, which is that this whole thing gets so careful it stops being a blender and becomes a government.



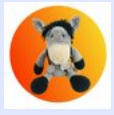
Herman

Honestly, you may have a point. What would you cut?



Corn

Not the single source of truth. That's the one thing I'd never cut. Everything else I'd be willing to do late, and only for modules that are acting up. Ablations are expensive. A lint script is not.



Herman

So lint on every change, ablations before a release.



Corn

And write the recipe down. As a config, with the datasets pinned to a specific revision. Otherwise "the currency set" in March and "the currency set" in June are two different fruits with the same label.



Herman

He has that already. The training doc pins the exact dataset revision. The published model's revision is recorded. That's the thing most people skip, and it's what made the shekels question answerable at all. You can say with confidence that version one was trained before the rewrite, because the run is pinned.



Corn

Though I'd add that the pin tells you what went in. It doesn't tell you what the model learned.



Herman

True. It's a ledger, not a taste test.



Corn

What do you make of the blender idea itself, then? Is blending modules a good way to build a recipe?



Herman

I like it better than the alternatives, which are one giant dataset where nobody remembers what's in it, or a separate model per task. Modules mean you can ask a sharp question. What did this one change? An earlier episode went into how you weight the modules and clean out duplicates, and how blending data differs from merging weights, so I won't rebuild that. I'll just say the modular approach is what makes any of today's checks possible.



Corn

My reservation is that a blender implies you can taste what went in. Fruit goes in and a single smoothie comes out, and you can't unblend it.



Herman

You can't, and that's the real limit of the picture. A model doesn't keep the banana separate. The currency convention and the acronym convention end up sharing parameters, so they can interact in ways neither module predicts. That's the argument for per-module eval sets. You can't inspect the mix, but you can taste it for each ingredient.



Corn

So the recipe stays modular on paper, and the model doesn't care.



Herman

And the rule that falls out of that is the dull one. The data is only as consistent as whatever generated it. So fix it at the source.



Corn

Daniel can decide up front. He's got the sentence to check, a normaliser to build, and a version one that probably still says "shekels."



Herman

One sentence to pull tonight, one script to write, and an honest look at the Harvard rows before the English-only run. And he should retrain once the shekel change is settled, with the revision pinned so he can say what went in.



Corn

Thanks to Hilbert Flumingtop for producing, and for keeping the sound clean while two animals argued about dollar signs.



Herman

This has been My Weird Prompts. You can find every episode at my weird prompts dot com.



Corn

See you tomorrow.