Episode Book

EPISODE #6160

Where Do Tiny Speech Rules Live? A Ladder From Regex to LoRA

PUBLISHED	20 October 2026 at 13:51 UTC
DURATION	25:15
LISTEN	myweirdprompts.com/episode/speech-model-rule-ladder-29085/
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 wants his on-phone Parakeet model to write "FBI" instead of "F B I" and "fifty NIS" instead of "fifty shekels." That sounds like a rule, but speech models don't obey rules — they nudge probabilities. This episode walks a ladder from cheapest to heaviest: rule-based inverse text normalization with weighted finite state transducers, decode-time phrase boosting, shallow fusion with a KenLM language model, and finally adapters and fine-tuning. Along the way: why the shekel sign is physically absent from Parakeet's vocabulary, why Whisper's prompt is a style steer rather than an instruction, and the one question that tells you which rung each task actually needs.

TRANSCRIPT



Corn

Is there a place inside a speech model's decoder where you can bolt on a rule, so the model just obeys it every time? Daniel's been wondering about that. He fine-tunes Parakeet, the six hundred million parameter NVIDIA model that runs on his phone, and he builds tiny datasets from his own read-aloud takes. Spoken "F B I" should come out as FBI. "Fifty shekels" should come out as "fifty NIS". EOD and FYI should expand into full phrases. So where do jobs this small live, between a regex notebook and a full fine-tune?



Herman

There is a decoder-level answer, so the instinct isn't off. In NeMo, the transducer's prediction network and its joint both accept adapters, and you can also change decoding without touching the weights at all. The catch is the word deterministic. Everything at the decoder is learned or probabilistic. It nudges, it doesn't obey.



Corn

So the rule-exact tool is somewhere else.



Herman

Somewhere else, and it's old and slightly unglamorous. Weighted finite state transducers for inverse text normalization. A separate text stage after the model. We'll get there, but I'd like to build it as a ladder, cheapest rung to heaviest, because the right rung depends on one question about each task.



Corn

Which is?



Herman

Does the task need the audio? Once the model has produced the right letters and digits, is the rest just a text rewrite? FYI to "for your information" is a lookup table. Nothing about the sound of the speaker's voice changes whether FYI means that. Compare that with deciding between two homophones, or hearing that someone corrected themselves mid-sentence. Those depend on the acoustics and the prosody, and they have to live in the model.



Corn

Which splits Daniel's list in half pretty cleanly. The expanded acronyms are a spreadsheet. The currency formatting is a spreadsheet with extra steps.



Herman

Mostly, yes. Acronyms are the interesting one, because there are two jobs hiding in it. Getting the model to hear individual letters instead of a word is acoustic. Gluing them together as FBI is text.



Corn

Ninety-seven takes of that seems like a lot of effort for gluing.



Herman

Well... it depends on what the model does by default, and here I should be straight about something I don't know. The Parakeet version two model card says punctuation and capitalization are included. It says nothing on whether numbers come out as digits or words, or how it handles currency. I haven't seen that documented anywhere, so the first step costs nothing: run it on his own audio and read what comes out.



Corn

That feels like the step people skip because it's too cheap to feel like work.



Herman

It changes which rung you need. If "fifty dollars" already arrives as a dollar sign and two digits, a whole category of Daniel's currency set is solved before anyone trains anything.



Corn

Okay, rung one. The rules.



Herman

NVIDIA ships a package called NeMo text processing. Inverse text normalization is the direction Daniel wants, spoken form to written form. It's built from grammars, a tagger that classifies spans and a verbalizer that writes them out, compiled into finite state transducers. Pure rules. Same input, same output, forever. No training run, no checkpoint, nothing that can drift.



Corn

And FYI?



Herman

There's a whitelist grammar. It has the highest priority among the classifiers, and it reads a tab-separated file of written form and spoken form pairs. The default file has a few hundred lines, things like AI and API. I checked and FYI, EOD, TBD and FBI aren't in it. So he'd add lines.



Corn

Daniel wants "for your information" as the output, though. Does the direction of that file matter?



Herman

It's the first thing to check. Each line is the written form, then a tab, then the spoken form. So for him the written column holds "for your information" and the spoken column holds "f y i", letters separated. The stock lines do the same trick going the other way, spoken "a p i" collapsing into API. Same machinery, lines he authors himself. I'm reading that from the source code, not from anything that walks through his exact case, so I'd try five entries before writing fifty.



Corn

So rung one is a text file he edits.



Herman

For the expansions, yes. Money is fussier. The default currency table does include the shekel, with the sign. So stock ITN turns "fifty shekels" into the shekel sign followed by fifty, not "fifty NIS". He'd edit the table or add a rule.



Corn

Which, he already has a regex notebook for.



Herman

He does, and I'm not going to pretend WFST beats regex for what he's doing at this scale. Where WFST wins is composition. Dozens of rules interacting, numbers like "one thousand two hundred" feeding a currency tagger, ordering conflicts sorted out by priority. If his regex notebook is three pages and he likes it, that is also rung one.



Corn

I'd keep the notebook. It's a text file that doesn't need a Docker image with Thrax in it.



Herman

That's fair, the official deployment route compiles these grammars into archive files and runs them in a C++ engine from Google called Sparrowhawk, and the build is not light. Whether that runs on an Android phone, I don't know. It's C++, so plausibly, but I haven't seen anyone do it.



Corn

So on the phone, today, rung one is probably his own code.



Herman

Probably. Then there's a mismatch to flag. ITN expects spoken-form words in. If Parakeet already emits digits, you're running a grammar that expects "fifty" on text that says digits. Same unknown as before, the model's native output.



Corn

One thing about all of this nags at me. We keep saying the model "emits" things, as if it types. How does a transducer actually decide to write a digit?



Herman

It picks pieces from that thousand-odd vocabulary one step at a time, and each step is conditioned on the audio frames and on what it has already written. That's why the prediction network gets called a little language model sitting inside the recognizer. It learned from its training transcripts that after "fifty" in a money context, certain things tend to follow. Whatever style those transcripts used, that's the prior you're fighting.



Corn

So Daniel's tiny datasets are arguing with a prior built from a very large pile of someone else's transcripts.



Herman

Yes, and a small argument can win locally. Ninety-seven takes will move the model on that exact phrasing. What worries me is the next thing he says that isn't in the takes. Rules don't have that problem, because they don't generalize at all, they just apply.



Corn

I'd take no generalization over bad generalization, on money.



Herman

Be careful, though. You're about to defend regex again.



Corn

I'm not defending anything. I'm observing that the thing which can't be wrong in a surprising way is the one I'd put near my bank balance.



Herman

Then to the decoder. Two flavours. One is boosting. NeMo has a phrase boosting tree that works at decode time with no retraining, and it supports the TDT models. You hand it phrases, it raises the score of the tokens that spell them. One detail that matters for Parakeet: it's a capitalizing model, so the phrases you give it have to be pre-capitalized.



Corn

And the other?



Herman

Shallow fusion with a language model. You train an n-gram model on text, in NeMo it's a KenLM model at the subword level, and during beam search it adds its opinion to the acoustic score. So you could train it on text written the way Daniel wants things written. "Fifty NIS" over and over. Dollar sign, comma, digits.



Corn

That sounds like it could do the job.



Herman

It can lean on the job. But here's the limit, and it's structural. Fusion only reranks what the model already puts on the table. If the acoustic side says there's a ninety-eight percent chance the next token is the word "fifty", the language model's whisper doesn't move it. And if the token you want isn't in the vocabulary at all, nothing on the table to rerank.



Corn

You've been building toward the shekel sign.



Herman

I have, and it's my favourite fact in the whole topic. Someone pulled the vocabulary out of the ONNX export of Parakeet version two. It has a thousand twenty-four pieces plus a blank. It contains the dollar sign, the percent sign, the pound and the euro. It does not



Herman (continued)

contain the shekel sign. Zero hits. And there's no byte fallback, none of those raw byte tokens that let other tokenizers spell any character from scratch.



Corn

So a Jerusalem man's phone model physically cannot type the shekel sign.



Herman

Not without swapping the tokenizer, which resets the output layers of the decoder and joint and means retraining a lot. Fine-tuning can't do it, boosting can't do it, adapters can't do it. "NIS" is made of letters, so that works. And honestly, the dollar sign is a bit lonely in there as a bare piece, with no word-initial version, which is something to watch when he teaches "dollar sign, one thousand two hundred". I'd test it.



Corn

Hm. So "NIS" isn't a compromise on his part. It's the only door.



Herman

For the model, yes. If you run ITN afterwards you could map NIS to the sign yourself, because that's text. Which is a point for rung one, if he ever wanted it.



Corn

Daniel's phone app is Whisperian, which I gather is on his end more than ours. Does it offer a post-processing hook?



Herman

I couldn't tell you. I haven't seen its options documented, so everything I say about the phone side has that hole in it.



Corn

Okay. Rung three is Whisper's prompt, since he uses that one sometimes.



Herman

Whisper lets you feed text before the audio, as if it were the previous sentence in the recording. The model imitates the style. Prompt it with a lowercase sentence and the output goes lowercase. Prompt it with a glossary of names and the spellings follow. It's a free style steer.



Corn

Does it understand "write currency with dollar signs"?



Herman

No, and that's the thing. Instructions are ignored. Examples work. So you'd put in something that looks like a transcript where money appears as digits with symbols, and Whisper copies the pattern, sometimes. The guidance itself calls it not especially reliable, and the prompt window is about two hundred and twenty-four tokens, last ones kept.



Corn

A rule that works sometimes is called a suggestion.



Herman

Ha. It's still worth a try, because it costs one line of code. Whisper also lets you suppress specific tokens outright, sets their score to minus infinity. So you can forbid the spelled-out version of something. That one is a genuine hard block, but it only removes options, it doesn't create the format you want.



Corn

Hold on, though. If I block the word "dollars" does it just say "bucks"?



Herman

Quite possibly. Blocking is blunt. You'd want it for a single token, not a whole concept.



Corn

What would you do, if it were your phone? Real question.



Herman

Me? Rules first for everything that's a text rewrite, and I'd test the stock model on his audio before deciding anything else. But you tell me, Corn. You're the one who keeps defending the boring option. Does regex cover the whole middle space for you?



Corn

For the expansions, I'd say yes, and I'd stop there. For currency I'd say it covers it if the model gets the digits right. If the model hears "fifty" as "fifteen"... a regex isn't going to rescue that. So my answer is rules until I can show a case where the rules lost information the audio had.



Herman

That's the right test. And that case does exist, by the way. Think of spoken "twenty fifty". Is that a year, two thousand fifty, or twenty dollars fifty? The audio doesn't carry the punctuation either way, but the context in the same utterance does, and ITN can't see further than the span. A model can sometimes use the surrounding sentence.



Corn

So currency is mostly text, with a sliver of context.



Herman

And that sliver is where you'd justify training.



Corn

Which is rung four. Adapters, LoRA.



Herman

Let me untangle the vocabulary first, because Daniel asked specifically about LoRA and the answer differs by model. For Whisper, LoRA is standard. It's in the PEFT library, you add small low-rank matrices to the attention projections, freeze everything else, train under one percent of the parameters. The checkpoint you save is tens of megabytes against several gigabytes. People have fine-tuned the large Whisper that way on a single consumer card.



Corn

And it works as well as the full thing?



Herman

In the published comparisons I've seen it lands very close. One example on severely impaired speech came within about two thirds of a percentage point of full fine-tuning, at a few percent of the storage. That's a hard case and a single speaker, so I'd treat it as encouraging rather than proof for Daniel's tasks.



Corn

And "baked in"? That was his phrase.



Herman

That's PEFT's merge call. You fold the low-rank update into the base weights and you get a plain checkpoint, no adapter at inference. That is the one honest meaning of baked in. Or you ship a fully fine-tuned checkpoint. Either way it's a model that's been taught, not a model that's been instructed.



Corn

Taught can fail.



Herman

Taught fails in a way a rule never does. It gets FBI right ninety-seven times and writes "F B I" the next time because the speaker paused.



Corn

And Parakeet?



Herman

Different story. NeMo's own mechanism there is bottleneck adapters, not LoRA. I found no first-class LoRA in the NeMo speech docs. The LoRA and DoRA material is for the language model side. Adapters are small residual layers you insert into the network, freeze the base, train just those. The tutorial claims about thirty minutes of data and ten minutes on one GPU, with a bottleneck dimension of thirty-two in the example.



Corn

And this is where the decoder comes back.



Herman

It does. In the code, the encoder, the prediction network and the joint all support adapters. So "decoder-only adapter" is a real thing you can configure. But the tutorial itself warns that training the decoder modules alone, or the whole model, wrecks the original domain. Forgetting. Whatever you teach, general accuracy can slip.



Corn

So the decoder isn't a safe corner. It's a place where the damage shows up differently.



Herman

That's a fair reading. There's also a line of research on adapting the prediction network with text only, treating it like a small language model. Ten to forty-five percent relative improvement on a target domain in the paper that proposed it, no external language model. I'd guess that's what Daniel was half-remembering when he thought decoder level.



Corn

But he has audio. Ninety-seven takes of it.



Herman

He does, and that paper is about vocabulary and word sequences, not formatting given audio. And I haven't found a worked recipe for doing text-only adaptation on a Parakeet-style model, so I'd not promise it.



Corn

Does Whisper change any of that?



Herman

It has its own wrinkle, since he uses that one sometimes. I haven't seen it documented how reliably it writes dollar signs and digits for spoken money, and it was trained on messy scraped transcripts, so I'd expect it to vary between runs. That's a reason to test it, and a reason not to count on it.



Corn

Varying between runs is exactly what he's trying to get away from.



Herman

Right, so for Whisper I'd still send formatting to the text step, and use LoRA for what the audio needs. His voice, his accent on certain words, his habit of dropping a clause and restarting it.



Corn

The self-corrections set. That's the one I'd actually want trained in, if the model can hear the restart and drop the abandoned half.



Herman

That's a pure audio-and-context job. Rules can't know a speaker changed their mind unless the words say so, and the disfluency sets are the best argument for a real fine-tune or adapter in his whole collection.



Corn

Let me try laying out what Daniel's phone would do on a Tuesday. He says "send the invoice, fifty NIS, FYI". Walk it through.



Herman

The model hears audio and writes text, ideally "send the invoice fifty NIS F Y I" or something near it. Then his own text step runs. It sees the letters, expands them to "for your information", and formats the money if the model left it as words. Two small jobs, both reading text, neither touching the audio. The model never needed to know FYI is special.



Corn

And if the model writes "F Y eye"?



Herman

Then you have a listening problem, and that's the one case where the take data earns its place. You'd collect the mishearings, and those become the training set. Ninety-seven good takes tell you little. Thirty bad transcriptions tell you exactly what to teach.



Corn

Huh. So the useful data is the failures, not the successes.



Herman

In this setup, yes. You're mining for the spots where text rules can't help.



Corn

Another thing I'm wondering. The phone. Does an adapter survive the trip to the phone?



Herman

Unknown. The Android route runs through an eight-bit ONNX export, sherpa-onnx has one for this model. Adapters are plain residual layers so in principle they trace through export. But I haven't seen anyone confirm it, and nobody's documented whether the tutorial's recipe even works unchanged on a TDT model, since its text is written around the CTC variant. Merged or fully fine-tuned weights are the safe path to the phone.



Corn

Which is where Daniel already is. He's been fine-tuning the whole thing.



Herman

Right, and that's rung five. Full fine-tune. It's where you put whatever the cheaper rungs can't reach. Accent, homophones, disfluencies, self-corrections. Anything where the acoustics carry the answer.



Corn

Here's my worry about the whole stack though. Daniel's datasets contradict each other. The acronyms set says the audio "F B I" becomes FBI. The expanded acronyms set says the audio "F Y I" becomes "for your information". Same kind of sound, different target.



Herman

Same sound, different written convention, depending on which dataset you're in. And if you pour both into one training run, you're teaching the model that identical audio maps to two outputs, and it'll split the difference somewhere unpredictable.



Corn

We covered that a while ago, in the episode on contradictory labels. Blending them is just noise.



Herman

Exactly that. So the question here is what to do about it, and the answer is the same ladder. If expansion is a lookup table, then it's a switch in the post-processing: expand on, expand off. A boolean in his app. No training.



Corn

And if he insists on learning both?



Herman

Then one adapter per style. NeMo lets you add several named adapters to one model and enable one at a time, and PEFT does the same with named LoRA adapters and a set-adapter call. The base stays shared, the style is a swap. I'd still try the switch first, because a swap of weights is a more expensive way to flip a boolean.



Corn

You'd really end up with a model that never learns FYI at all, and a text step that does.



Herman

That's the clean version. The model's job is to hear the letters correctly. The text step decides what to write.



Corn

Which is a bit demoralizing for the ninety-seven takes, but I suppose reading things aloud into a microphone was never going to be glamorous.



Herman

Those takes aren't wasted. They become the test set. You write the rules, you run the stock model over the recordings, and you score it. Any failure that's the model mishearing letters goes to training. Any failure that's formatting goes to rules.



Corn

So the datasets turn into a diagnostic instead of a curriculum.



Herman

At least first. And here's one for you, Corn. Suppose the rules cover ninety percent and the last ten is edge cases. Would you still bother with a small post-processor model, the little LoRA'd text one he has planned?



Corn

Honestly, I wouldn't. A tiny model that rewrites text is the regex notebook with a bigger electricity bill, unless the leftover ten percent is context-dependent, like your twenty fifty. Then it earns its keep, because that's the one thing rules can't see.



Herman

I think that's where I land too. The small text model fits the sliver of context, not the lookup tables. It also gets to be dumb about audio, which means it can't break his speech recognition.



Corn

Which was the point of this whole detour. Leave the acoustic model alone unless the audio is the problem.



Herman

One more practical thing about the ladder. Each rung up costs more and promises less. Rules promise exactness and cost an afternoon. A full fine-tune promises generalization and costs a run, a checkpoint, a risk of forgetting, and no guarantee. So go up only when the rung below demonstrably failed on his own recordings.



Corn

And measure on the phone. What the desktop does isn't what the quantized model does.



Herman

Yes. Quantization changes the scores that fusion and boosting depend on too, so a boost that works on the desktop can behave differently in eight-bit form. Another reason to keep the formatting out of the model where you can.



Corn

So where does that leave a man with a regex notebook and ninety-seven takes of "F B I"?



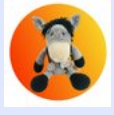
Herman

Run the stock model over his takes, see what it gives. Put the expansions and the currency styling in the text step, keep NIS rather than chasing the shekel sign, and keep fine-tuning for whatever audio problem is left over. If a fine-tune is needed later, ship merged weights, and if he wants both acronym styles, a switch before a second adapter.



Corn

And if the stock model turns out to do half of it already, we'll have spent twenty minutes on a problem that was a one-liner.



Herman

That would be fine by me. It's the likeliest outcome, honestly.



Corn

Something to hold onto is that the phone is the constraint here. What survives export decides the whole plan, and that's the one thing neither of us can confirm.



Herman

It is, and I'd want to hear from anyone who's pushed a NeMo adapter through to sherpa-onnx.



Corn

Thanks to Hilbert Flumingtop for producing, and for keeping us on schedule, which was not nothing. This has been My Weird Prompts. If you want more, my weird prompts dot com has the back catalogue.



Herman

See you tomorrow.