Episode Book

EPISODE #5624

One Forward Pass: Building a Bounded Classifier

PUBLISHED	28 September 2026 at 14:20 UTC
DURATION	21:52
LISTEN	myweirdprompts.com/episode/bounded-classifier-fine-tuning-guide/
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

A bounded classifier emits a probability distribution over a fixed label set in one forward pass — no token generation, no JSON parsing, no helpfulness. This episode walks through what it takes to build one for a real pipeline: which encoder to fine-tune from (ModernBERT, DeBERTa-v3, or Liquid AI's LFM2.5 encoders), what size makes sense for a few thousand examples, and why a 150 MB weight file running on a laptop CPU can outperform a decoder LLM at a fraction of the cost. Then the harder questions: multi-label output with independent per-label thresholds, the zero-tag case that falls out of the arithmetic for free, and what to do when the taxonomy changes.

TRANSCRIPT



Corn

The model returns every label score in one forward pass. There is no text generation. There is no output parsing. That single sentence is the whole reason this episode exists.



Herman

And it's the sentence that explains why our own tagging pipeline has been the flakiest part of the machine for as long as it's existed.



Corn

Retired more than once. Never the part you could trust.



Herman

Daniel's noticed.



Corn

Here's what he wrote in. He wants to actually build a bounded classifier for this show. Tomorrow, if he felt like it. He's got a defined channel list, a defined tag list, and he wants to add a third thing, an episode-type classifier, general episode versus the occasional Q and A format.



Herman

Which is a separate enum, for the record. He should not fold that into the tag list.



Corn

Noted, and we'll get there. He's got more than five thousand published episodes of training data, most of it already tagged by a language model, imperfectly. He'd either hand-tag about a hundred episodes to establish the pattern or take the existing automatic tags and hand-edit them, and he figures the difference shouldn't materially affect the classifier.



Herman

He's probably right about that.



Corn

Then four questions. One, what's the most logical base model to fine-tune from, which is his way of asking whether there's a standard baseline classifier people regard as a good foundation. Two, what size should the custom classifier be, in parameters and in weight-file size. Three, how do you handle multi-label output, nullable tags, and the case where zero tags apply. Four, when the taxonomy gets revised, how do you fine-tune on the amended list, or is that approach better avoided.



Herman

That last one is the one that eats people.



Corn

So before we answer any of it, we need to be precise about what a bounded classifier actually is, because it is not the same thing as asking an LLM for JSON.



Herman

Right, and this is the distinction that gets blurred constantly. A bounded classifier is a model that emits a probability distribution over a fixed label set in a single forward pass. That's it. The architecture itself has nowhere else to go. The output layer has one slot per label, and the model fills those slots with scores.



Corn

Whereas the JSON-schema approach, the constraint is applied at decode time. The model is still autoregressive. It's still generating tokens one at a time. The schema is a mask sitting over a generative process.



Herman

It's a mask, not a replacement. And the closest the LLM side gets to a real guarantee is grammar-constrained decoding, where you mask out any token that would violate a context-free grammar. That does guarantee syntactic validity. You will get well-formed JSON.



Corn

You'll get well-formed JSON that's wrong.



Herman

Park and colleagues at NeurIPS two years back put it better than I can. Grammar-constrained decoding can distort the model's distribution, so you get outputs that are grammatical but appear with likelihoods that aren't proportional to the ones the model actually assigned. Which means low-quality. That's the finding. The grammar didn't make the model better at the task. It made the model worse at the task while making the output prettier.



Corn

So the constraint is fighting the architecture.



Herman

Constantly. And one more thing worth saying out loud, because listeners will go looking. Bounded classifier is not a term of art. It's not in the literature. There's no canonical definition you'll find on arXiv. The concept maps onto encoder-based multi-label classification on the model side, and grammar-constrained decoding on the LLM side, but the phrase itself is ours.



Corn

Good. So nobody emails us asking where the paper is.



Herman

Nobody emails us asking where the paper is.



Corn

Then let's take the first question, because it's the one with a clean answer. If you're building a small proprietary classifier for your own pipeline tomorrow, what do you fine-tune from?



Herman

The consensus modern baseline is encoder-only. Not decoder. And the framing from the ModernBERT launch is blunt about why. Decoder-only models are too big, slow, private, and expensive for many jobs, and you don't want to pay prototype prices once you're in mass production.



Corn

That's the line.



Herman

And the companion observation is the one that should reframe how people think about their own stacks. Whenever you see a decoder-only model in deployment, there's a reasonable chance an encoder-only model is also part of the system. The converse is not true. Encoders hide inside pipelines that are nominally built around an LLM.



Corn

Because the LLM is doing the thing that needs judgment and the encoder is doing the thing that needs to happen ten thousand times.



Herman

Exactly that division of labor. And the download numbers back it up. BERT alone is the second most downloaded model on Hugging Face, more than sixty-eight million monthly downloads. Encoder-only models in total pull over a billion downloads a month.



Herman (continued)

Decoder-only models pull about three hundred and ninety-seven million. Encoders are not the legacy option. They're the workhorses.



Corn

So what are the actual candidates for Daniel's task.



Herman

Three real ones. ModernBERT-base, which is a hundred and forty-nine million parameters, eight thousand one hundred and ninety-two token context, and it's described as a slot-in replacement for any BERT-like model. It's the first base-size model to beat DeBERTaV3 on GLUE while using less than a fifth of DeBERTa's memory.



Corn

Second.



Herman

DeBERTa-v3. The long-standing favorite in Kaggle and production. It still wins on precision-heavy tasks. There's a 2026 result from the PAN workshop where DeBERTa-v3-large scored 0.882 against ModernBERT-large's 0.96, so it's not a clean sweep either way. Depends on the task.



Corn

And third.



Herman

Liquid AI's LFM2.5 encoders. Two hundred and thirty million and three hundred and fifty million parameters, both eight thousand one hundred and ninety-two token context, released at the end of July. And they're positioned explicitly for classifiers, intent routers, and safety filters. That's the pitch. Not general language understanding. The narrow jobs.



Corn

Which is exactly Daniel's job.



Herman

Exactly Daniel's job. And the Liquid framing on where these run is the part that matters for a podcast pipeline. Classifiers and intent routers run constantly, often on CPUs rather than GPUs. They're not the thing you spin up a rented accelerator for. They're the thing that runs on the box you already have.



Corn

So what size should he actually build.



Herman

For a taxonomy with a handful of labels and five thousand episodes, base size. A hundred and forty-nine to three hundred and fifty million parameters. There is no evidence that a decoder LLM is the right foundation for this, and there's a lot of evidence pointing the other way.



Corn

Give me the weight-file arithmetic, because that's the question people actually want answered.



Herman

It follows straight from the parameter count. At full thirty-two-bit precision you're at about four bytes per parameter, so a hundred and forty-nine million parameters lands around six hundred megabytes. At sixteen-bit, two bytes per parameter, you're at about three hundred megabytes. At eight-bit you're down around a hundred and fifty.



Corn

So a hundred and fifty megabyte file for the small end.



Herman

A hundred and fifty megabyte file that runs on a laptop CPU. That's the whole thing. That's the classifier.



Corn

The comparison that should end the argument for anyone still on the fence is the filtering cost. Fine-tuned BERT filtering fifteen trillion tokens came in around six thousand H100 hours, roughly sixty thousand dollars. The decoder-only equivalent for the same job was over a million.



Herman

Over a million dollars. Same task. And that's not a tuning difference, that's an architecture difference. You're paying for generation you don't need.



Corn

There's a CPU speed number too, isn't there.



Herman

There is, and it's the one that surprised me. At eight thousand one hundred and ninety-two tokens, ModernBERT-base takes over a minute and a half per forward pass on CPU. LFM2.5-Encoder-230M does the same pass in about twenty-eight seconds. That's roughly three point seven times faster, and it's the smaller model winning.



Corn

Which is counterintuitive until you remember that attention cost isn't linear in parameter count.



Herman

Right, and I'll be honest, I don't know the full architectural reason Liquid gets that gap. I know it's real and I know it's measured on the same context length, but I couldn't tell you which specific design choice buys it.



Corn

Fine. What's a concrete result for a small encoder on a multi-label task, so people know what good looks like.



Herman

The Liquid cookbook runs LFM2.5-Encoder-350M on a legal document benchmark, European Court of Human Rights cases, multi-label. After per-label threshold tuning it hit 0.8060 validation micro-F1. Test set was 0.7913 micro-F1, 0.7062 macro-F1, and 0.8400 micro average precision.



Corn

Point eight micro-F1 on a multi-label legal task with three hundred and fifty million parameters.



Herman

On a CPU.



Corn

So that's the architecture and the model choice. But Daniel asked two more questions, and they're the harder ones. Multi-label output and taxonomy revision.



Herman

The multi-label part is where the encoder approach stops being merely cheaper and starts being structurally better. The standard setup is not softmax over mutually exclusive classes. It's one linear output per label, trained with binary cross-entropy.



Corn

Which means each label gets its own independent probability.



Herman

Each label gets its own independent probability. So an episode can be technology and DIY at the same time, which is what Daniel wants, because the model isn't being forced to pick a winner. And here's the part that answers his nullable question directly. If every label scores below its threshold, the output is an empty list. That's not an edge case you engineer. That's just what the arithmetic produces.



Corn

The zero-tag case is free.



Herman

Which is worth sitting with for a second, because it's the exact failure that has burned everyone who's tried to get an LLM to return an empty array under a JSON schema. You end up writing prompt language about when to return nothing, and the model returns a tag anyway because it wants to be helpful. In a bounded classifier, there's no helpfulness. There's a threshold.



Corn

So where does the actual accuracy come from, if not the model.



Herman

Thresholds. That's the honest answer. The Liquid cookbook compares three setups. A fixed 0.5 threshold across everything, one tuned global threshold, and tuned per-label thresholds. And it picks the best checkpoint by validation average precision, which is threshold-independent, so you're not tuning the threshold and the model at the same time.



Corn

And the gap between global and per-label.



Herman

The classivore pipeline reports plus five percent F1 macro from per-category threshold optimization over a global threshold. Five points of macro F1, from changing the decision rule, not the model.



Corn

That's not a rounding error.



Herman

That's the difference between a classifier you trust and a classifier you retire. And it makes sense once you think about label frequency. If one of your channels shows up on forty percent of episodes and another shows up on two percent, a single threshold is going to be wrong for at least one of them.



Corn

There's a counter-argument on thresholds though.



Herman

There is. RAPT argues global thresholds are brittle and hard to maintain as document formats evolve, and proposes retrieval-augmented thresholds that are per-label and per-instance. I'd call that the frontier rather than settled practice. For Daniel's scale, per-label static thresholds are almost certainly enough. But it's a real open question whether they hold as the corpus drifts.



Corn

Now the fourth question. Taxonomy revision.



Herman

This is the one where the research is useful, because it tells you where the failure actually lives. The general phenomenon is catastrophic forgetting. After new tasks are learned, performance on old tasks degrades. That's well established.



Corn

But the mechanism is more specific.



Herman

The mechanism is more specific, and this is the finding I'd underline. The classifier causes the forgetting. Changes in the relative position between the class embeddings in the classifier and the features extracted by the language model lead to poor performance on old tasks even when the language model itself doesn't forget.



Corn

So the backbone is fine. The head is the problem.



Herman

Which is good news, because the head is a linear layer. It's the cheap part. So the practical answer to Daniel's question is, keep the encoder backbone, swap or retrain the head on the full amended label set, and re-tune the thresholds.



Corn

Full amended set, not just the new labels.



Herman

Full amended set. Incremental fine-tuning on only the added labels is where people get burned. There's a project called adaptive-classifier that does dynamic class addition and continuous learning, prototype memory plus an adaptive neural layer. And its own benchmarks show the adaptation is imperfect. Router success rate on high-cost routes dropped from 40.71 percent to 29.59 percent after adaptation.



Corn

It got worse.



Herman

It got worse at the thing it was already good at. Which is the forgetting, showing up in the numbers.



Corn

So the recommendation is retrain the head on everything, every time the taxonomy changes.



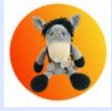
Herman

Every time. And it's cheap enough that there's no reason not to. The classivore estimate for a seven-hundred-category taxonomy over thirty thousand pages puts labeling at fifteen to twenty-five dollars using the batch API, and training on a single 4090 at about forty-five minutes.



Corn

For seven hundred categories.



Herman

Daniel has a handful of channels, a handful of tags, and one binary episode-type flag. This is a weekend. It's not a quarter.



Corn

So the picture is, encoder backbone, base size, per-label thresholds, retrain the head when the list changes.



Herman

That's the picture. And the whole thing fits in a hundred and fifty to three hundred megabytes and runs on the CPU that's already there.



Corn

Which leaves one thing we haven't talked about, and it's the thing I keep circling back to. Every one of those answers assumes the label list is stable enough to be worth training against.



Hilbert

The list is never stable. I did a stint as a contractor at an e-commerce company, and my whole job was maintaining the category tree the product classifier was trained on. That was the title. Taxonomy wrangler.



Herman

How many categories.



Hilbert

Started around four hundred. Ended around eleven hundred. Marketing added a New Arrivals category in the spring, and it overlapped with everything, because everything is new at some point. So the classifier learned it, and it fired on anything with a recent date field, which was most of the catalog.



Corn

Did it get retired.



Hilbert

It got retired in the fall. And the label stayed in the model. It kept firing on things for another year, low confidence, but above threshold, because nobody retrained the head. They just stopped showing it in the interface.



Corn

So the dead label is still in there.



Hilbert

Still in there. And the thing I'd tell Daniel is that the Miscellaneous category accounted for about forty percent of all products by the time I left. Every new category created a new edge case, and every edge case got dumped in Miscellaneous, and Miscellaneous was a real category in the training data, so the model learned to use it generously.



Herman

That's the zero-tag problem wearing a hat.



Hilbert

The model can't tell the difference between nothing applies and I don't know, so it picks the bucket that means both. And once that bucket exists in your taxonomy, it grows.



Corn

So the model choice is almost beside the point.



Hilbert

The model choice is a rounding error. You can retrain the head in an afternoon. You cannot retrain the four people who decide what the categories should be. That's the part that costs you. I had a spreadsheet of every category addition and who requested it, and I could tell you which ones were going to be dead within a year, because they were named after a promotion.



Corn

Did anyone ever ask you.



Hilbert

Once. I said the New Arrivals category would cannibalize everything. They added it anyway. It's not a technical problem. It's a governance problem, and the classifier just inherits whatever governance you've got.



Corn

The model is the easy part.



Hilbert

The taxonomy is the product.



Corn

So if the taxonomy is the product, what does a well-maintained one for five thousand episodes actually look like.



Herman

The honest answer is I don't know, and I don't think anyone does, because it depends entirely on how often the content shifts. But there's a structural point underneath it. The encoder approach makes the model cheap enough to retrain frequently, which means the bottleneck moves. It stops being a compute question and becomes a human question about who decides what the labels are and how often they're allowed to change.



Corn

And the frequency question is real. Revise too rarely and the taxonomy stops describing the show. Revise too often and you're retraining the head every month and re-tuning thresholds every time, and every revision is a chance to introduce a label that overlaps with three others.



Herman

Which is the New Arrivals problem. A label that sounds useful and is actually a bucket.



Corn

So the discipline is, a new label has to be disjoint from every existing label, or it's not a label, it's a tag.



Herman

That's a clean rule. Channels are disjoint. Tags can overlap. And the nullable case belongs to tags, not channels, because an episode always has a channel even if it's a bad fit.



Corn

Which means Daniel's episode-type classifier should be its own enum, disjoint from both.



Herman

Its own enum, disjoint from both. Two values. General and Q and A. Don't put it in the tag list.



Corn

And the retraining cadence falls out of that. Channels almost never change, so the channel head is basically static. Tags drift, so the tag head gets retrained on whatever schedule the tag list actually changes. And the thresholds get re-tuned every time, because a new label changes the calibration of the ones around it.



Herman

That's the maintenance regime. And none of it is expensive. It's just a decision somebody has to own.



Corn

Which is a different kind of problem than the one Daniel started with. He came in asking about base models and parameter counts, and the answer to all of that is, base-size encoder, a hundred and fifty to three hundred megabytes, per-label thresholds, retrain the head. That's a weekend of work. The part that will actually determine whether this classifier survives is whether the label list is treated as a document with an owner or as something that gets edited whenever someone has an idea.



Herman

And the encoder makes that harder to ignore, in a good way. When retraining is cheap, you can't hide behind the cost of retraining. You have to actually decide.



Corn

There's a version of this that's even more uncomfortable, which is that the encoder gives you a measurement you didn't have before. If you retrain the head every time the taxonomy changes, you can see exactly which label additions moved the macro F1 and which ones didn't. That's a feedback loop the LLM approach can't give you, because the LLM's outputs aren't calibrated enough to compare across taxonomy versions.



Herman

So the taxonomy stops being a matter of taste and starts being something you can actually evaluate.



Corn

It starts being something you can actually evaluate. Which means the person who owns it has to defend their additions with numbers instead of vibes.



Herman

That's the part that will make people resist the whole approach.



Corn

It will. But it's also the part that makes the classifier survive past the second year.



Herman

Which is the only timeline that matters.



Corn

If you're building something like this, or if you've maintained a taxonomy that got away from you, we'd like to hear about it. Reviews help other people find the show.



Herman

Our producer, Hilbert Flumingtop, keeps the whole thing running.



Corn

This has been My Weird Prompts.



Herman

We'll be back soon.