Episode Book

EPISODE #5621

Constrained Decoding for AI-Generated Podcast Tags

PUBLISHED	28 September 2026 at 11:59 UTC
DURATION	22:55
LISTEN	myweirdprompts.com/episode/constrained-decoding-podcast-taxonomy/
HOSTS	Corn, Herman
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:38 UTC

SYNOPSIS

Daniel's AI-generated show deploys to Vercel, runs on serverless Postgres, and produces exactly the kind of wide-ranging content he wants. The problem is the feed — automatic tagging that started inventing its own categories, then duplicating them in ways nobody notices. This episode works through the fix: why unconstrained categorization never converges, how JSON Schema enums and context-free grammars make label bloat structurally impossible, and why a fine-tuned encoder beats a frontier model for a stable label set at one to two orders of magnitude lower cost. We also cover the counterargument — that a confident wrong classification is worse than no classification — and the read-at-classification-time pattern that dissolves the serverless redeploy problem entirely.

TRANSCRIPT



Corn

Okay. I'm going to be honest, I read this one twice.



Herman

Twice?



Corn

Twice. Not because it was confusing. Because it's the most Daniel prompt we've ever gotten. He's built a whole pipeline, it works, and now he wants to talk about the part that's quietly falling apart.



Herman

The tagging.



Corn

So, here's what he wrote. The show is an AI-generated pipeline. It deploys to Vercel, it runs on serverless Postgres, the model was picked to fit the pipeline rather than the other way around. And the part he cares about most, his words, is generating a wide variety of content about a rich diversity of topics. That's the whole point of the show for him. So he wants to wrap it into a graph-based model for learning, and the front door of that is the website and the feed, which is what listeners actually touch.



Herman

And the feed is where the taxonomy shows up.



Corn

Right. He experimented with automatic tagging. A model defines categories and tags, and every time a new episode comes out, a small model looks at the list and picks the best fit. He calls them channels now. Top-level themes. Tech, DIY, that kind of thing, so you can build a feed for people who only want one of those.



Herman

And that's the constrained version.



Corn

That's the constrained version. But the first thing he tried was the naive version, and he says it failed in the wild. The model could choose a category, and if it didn't exist, it just made one. Severe category bloat. Non-obvious duplication.



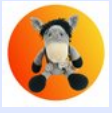
Herman

Non-obvious is the interesting word.



Corn

It is. And then the part I think is the actual heart of this. He says curating episodes into channels and tags is an editorial decision he wants to own personally. Deciding what tag a conversation gets, and even the shape of the taxonomy, whether there's an episode type at all, that's an editorial reflection of how he thinks the product should be consumed. So he wants the AI doing the classification, but he wants to draw the boundaries himself, and periodically review them. Rename tags, update topics, refine the set.



Herman

And then the propagation problem.



Corn

He wants a review backend where his edits push an updated list of categories and tags and types down to the agent that does the classifying. And he flags the wrinkle himself, which is that admin backends on serverless are messier than traditional ones, because sometimes you need a new deployment just to push a change.



Herman

He's right about that, and also wrong about it, and both of those are load-bearing.



Corn

That's a good tease. So where do we start?



Herman

Start with the vocabulary, because everything else hangs off it. Unconstrained categorization is an open-vocabulary generation task. The model is authoring labels. Constrained categorization is a closed-set selection task. The model is picking from a label space somebody else defined. Those are not two settings of the same dial. They're different jobs.



Corn

And the first job never converges.



Herman

It can't. Every run is allowed to invent. So the label set only grows. It's a ratchet. And the growth isn't random, it's patterned, which is what makes it so annoying. You get synonym drift and you get granularity drift.



Corn

Define those.



Herman

Synonym drift is four labels that mean one thing. Episode one gets tagged DIY. Episode two gets tagged Home Projects. Episode three gets Maker. Episode four gets Build-it-yourself. Now you have four channels describing the same theme, and an episode filed under any one of them is invisible under the other three.



Corn

Which is worse than having one bad channel, because now the content is scattered.



Herman

It's scattered and it's scattered invisibly. Nobody files a bug about a channel that shouldn't exist. They just don't find the episode.



Corn

And granularity drift?



Herman

Same disease, other axis. Tech spawns AI. AI spawns AI Agents. AI Agents spawns Agentic Workflows. Now you've got four levels of nesting, and no rule about which level an episode belongs at. Some episodes sit at Tech. Some sit at AI Agents. Two episodes that are basically the same kind of thing end up at different depths because the model felt like it that day.



Corn

And the non-obvious duplication he hit is just the predictable output of that.



Herman

It's the signature. Labels that look distinct on the page and overlap in meaning. A human reading the channel list wouldn't spot it immediately, which is exactly why it survives. It's not a model failure. The model did what it was asked. It was asked the wrong thing.



Corn

So the fix isn't a better prompt.



Herman

The fix is not a better prompt. You can write the most beautiful prompt in the world, and it's still a request. It's still asking. What you want is enforcement.



Corn

Say more, because I think people hear "constrained" and think it means "we told it nicely."



Herman

It doesn't mean that. It means the output is structurally incapable of being out of set. You define the channel list as a JSON Schema enum. The model's decoder is constrained so that the only tokens it can emit are the ones in your list. It physically cannot produce a label you didn't define. Bloat stops being a discipline problem and becomes an impossibility.



Corn

There's a line from the Anthropic context-engineering discussion on Hacker News that puts it well. Steering with grammar. Structured output with JSON schema, or context-free grammars directly, is a huge win. That's the move.



Herman

And it's the single most important implementation detail in this whole build. Everything else is plumbing. That's the load-bearing wall. If you get that right, the taxonomy can't drift. If you get it wrong, you're writing prompts forever.



Corn

Okay, so that's the mechanism. Now the part I actually want to push on, because Daniel's instinct was to use a small model, and I want to know if that's right or if it's just cheaper.



Herman

Both, and the research backs him. There's a benchmark from this year on cost-aware model selection for text classification. It studies exactly this shape of problem, structured classification with a fixed label space. And the finding is that fine-tuned encoder models, the BERT family, match or beat prompting a large language model, at one to two orders of magnitude lower cost and latency.



Corn

One to two orders of magnitude.



Herman

For a fixed label set. And the same paper has a line I'd frame and put on the wall. Indiscriminate use of large language models for standard text classification workloads can lead to suboptimal system-level outcomes. That's a polite way of saying you're paying frontier prices for a solved problem.



Corn

So Daniel's small model isn't a compromise. It's the correct answer.



Herman

For the classification step, when the label set is bounded and stable, yes. And that's the caveat. Bounded and stable. If your taxonomy is churning every week, a fine-tuned encoder is a bad fit because you'd be retraining constantly. If it's stable, the encoder wins on every axis that matters.



Corn

Which loops back to the editorial point. Daniel wants to own the boundaries. If he owns them well, they're stable, and the cheap reliable model is available to him. The discipline pays for itself twice.



Herman

That's a nicer way of putting it than I had.



Corn

Don't get used to it. Now, the counterpoint, because there is one and it's good.



Herman

There is. A production engineer on Hacker News, working on the Ghost macOS app, made an argument I keep thinking about. Deterministic routing beats LLM intent classification for the critical fork. His point is that an LLM classifier gets you to about ninety-five percent. And ninety-five percent sounds great until you ask where the five percent lands.



Corn

And it lands in the place you can least afford it.



Herman

His words. Quietly guessing wrong is worse than doesn't guess.



Corn

That's the whole argument in one line. A classifier that says "I don't know" is useful. A classifier that confidently files your DIY episode under Tech is a silent failure that nobody catches for six months.



Herman

So for taxonomy, that argues for a fallback. An Uncategorized bucket that actually surfaces for human review, rather than a model that force-fits every episode into something. You'd rather have a queue of ten unresolved episodes than a channel that's quietly wrong.



Corn

And that connects to the review loop, which is the thing Daniel actually asked about. Because if you're going to have a human reviewing, you need to be smart about what the human reviews.



Herman

There's a pattern for that. A validation layer. AI generates, automated validation catches type and format errors, a second model does adversarial review, and then the human only reviews the flagged items plus a random sample.



Corn

So you're not checking everything. You're checking exceptions.



Herman

The claim from the writeup is that it cuts review time by about eighty percent. And it's the right shape regardless of the number. Humans are expensive and they're inconsistent. Spend them on the edges, not the middle.



Corn

There's also a tool worth mentioning, BAML, which came up in a Hacker News thread about structured outputs. Some people find it better than raw JSON schema, which can be slow and sluggish to work with.



Herman

Worth evaluating, not worth committing to blind. If you're already comfortable with schema-constrained generation, you may not need it. If you're fighting the ergonomics, it's a real option.



Corn

Okay. That's the mechanism. The bloat, the constrained decoding, the model choice, the fallback. Now let's talk about how you actually build this, because that's what he asked for and I don't want to leave him with a philosophy lecture.



Herman

The architecture. Start with the data model, because everything else is downstream of it.



Corn

Postgres.



Herman

Serverless Postgres. A channels table. Id, name, slug, description, active flag, sort order. A tags table, same shape. An episode_types table. An episodes table with foreign keys into those. And a classification_runs table, which is the one people forget.



Corn

What goes in it?



Herman

Episode id, which model ran, which channel it chose, a confidence score, a timestamp, and a taxonomy version. That last column is the one that earns its keep.



Corn

Because it lets you reconstruct what the label set looked like when a given episode was classified.



Herman

If you rename a channel in March, you can still tell which episodes were classified against the old name. That's reproducibility, and it's also a history of your own editorial thinking.



Corn

Which is a nice thing to have for a show about curiosity. Your taxonomy becomes a record of what you thought the show was about, at a point in time.



Herman

Now the propagation question, which is the actual hard part Daniel flagged. How does an edit in the admin backend reach the classification agent?



Corn

Three patterns.



Herman

Three. The first one, and the one I'd recommend at this scale, is read-at-classification-time. The agent queries the channels and tags tables at the moment it runs. It builds the enum dynamically from whatever's in the database right then, and classifies. No deployment involved. You edit a row, the next episode picks it up.



Corn

So the taxonomy lives in data, not in code.



Herman

The taxonomy lives in data. And that single decision is what dissolves the serverless constraint Daniel was worried about. You don't need to redeploy to change a channel name. You need to write a row.



Corn

Which is the thing he half-knew and half-doubted. He said serverless admin backends are messier than traditional ones because sometimes you need a new deployment to push updates to the front end.



Herman

He's right about the front end and wrong about the data. UI code deploys rarely. Taxonomy data changes constantly. Those are two different clocks, and you should never couple them. Build the admin as routes and API handlers that read and write Postgres. You redeploy when you change the admin UI itself. You do not redeploy when you rename a channel.



Corn

And the second pattern?



Herman

Versioned taxonomy plus cache invalidation. You keep a taxonomy version integer. The classifier caches the label list, but it checks the version before it trusts the cache. When you edit the taxonomy in the admin, you bump the version. It's the same read-at-runtime idea with a cache in front of it, and it buys you the reproducibility from the classification_runs table.



Corn

When would you reach for that over the simple version?



Herman

When you're running enough episodes that you don't want a database round trip on every classification, or when you care about being able to say which label set was live at a given moment. For a daily show, honestly, the simple version is fine. But the version column costs you nothing and answers questions later.



Corn

And the third?



Herman

Build-time injection. You bake the enum into the deployed function. That's the one that does require a redeploy. And it's the wrong tradeoff for an editorial tool, because the whole point is that you want to edit this thing frequently. Only do it if the taxonomy is static, which for Daniel it isn't.



Corn

So he was describing a real constraint, and the answer is that he doesn't have to live inside it.



Herman

He doesn't. The constraint is real for code. It's not real for data. And the taxonomy is data.



Corn

Let's talk about the serverless Postgres side, because there's a trap there that people hit.



Herman

Connection pooling. Serverless functions open a lot of short-lived connections, and a Postgres instance will run out of them. You want something doing PgBouncer-style pooling in front, which the managed serverless Postgres options handle for you. Neon, Vercel Postgres, that family. It's not glamorous, but it's the thing that breaks first under load.



Corn

And the front end. If the site caches taxonomy-derived pages, an edit in the admin needs to reach the public site.



Herman

On-demand revalidation. You trigger it from the admin save action. In Next.js terms that's revalidating a path or a tag. The point is that the save button does two things. It writes the row, and it tells the front end to stop serving the stale version. No full redeploy.



Corn

Which brings us to the feeds, because that's the payoff. The whole reason he wants channels is per-channel feeds.



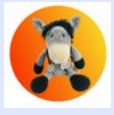
Herman

And per-channel feeds are just separate RSS documents filtered by channel. It's a well-trodden pattern. The constraint to remember is that Apple Podcasts mandates RSS 2.0 with an enclosure element per episode. So each channel feed has to be a valid feed in its own right, not a filtered view that breaks the schema.



Corn

And there's a practical note about feed size.



Herman

Clip them. Feeds are typically clipped to the latest N items rather than serving ever-growing XML. If you've got a daily show and you spin up six channel feeds, you don't want each one carrying the entire archive. Clip to the recent window and let the site handle the deep archive.



Corn

Okay. Now the editorial design question, because Daniel raised it directly and I think it's the most interesting thing in the prompt. Do you want an episode type?



Herman

You do, and you want it as a separate axis. Channel is the theme. Tech, DIY. Episode type is the format. Interview, deep dive, question and answer. Two separate constrained enums, not one flat list.



Corn

Why does that matter?



Herman

Because folding them together is exactly how you get granularity drift. The moment one list has to carry both "what is this about" and "what shape is this," the model has to guess which dimension you care about, and it'll guess differently on different days. Two enums, two questions, no ambiguity.



Corn

And it's an editorial statement. Saying "we have episode types" is saying the format is part of how you want people to browse.



Herman

It's a claim about the product. Which is why it should be Daniel's call and not the model's.



Corn

That's the through-line here, actually. Every technical decision in this build is downstream of an editorial one. The constraint is editorial. The axes are editorial. The review queue is editorial. The model is just the thing that fills in the blanks you drew.



Herman

And that's why the deterministic argument lands the way it does. The places where you can least afford a wrong guess are exactly the places a human should own. Taxonomy boundaries are that place.



Corn

So to Daniel's actual question. Nuts and bolts. Taxonomy in Postgres. Classifier reads it at runtime and builds the enum dynamically. JSON Schema enum for enforcement. Small fine-tuned encoder for the classification itself, with a confidence score. Low confidence routes to an Uncategorized queue. Admin is routes and handlers writing to the DB, with revalidation triggered on save. Version the taxonomy so you can reconstruct history. Two orthogonal axes, channel and type. Per-channel RSS feeds clipped to a recent window.



Herman

That's the whole thing.



Corn

It's a lot of plumbing for something that sounds like it should be a dropdown.



Herman

It always is.



Corn

Herman, I want to ask you something before we go further. You've been describing this like it's settled. Is any of it actually uncertain?



Herman

The revalidation specifics I'd verify against current docs before building. I'm confident about the shape, less confident about the exact API surface today. And the cost numbers from that benchmark are one paper. The direction is right, the precise multiplier I'd treat as indicative.



Corn

Fair. I'd rather hear that than a confident wrong answer.



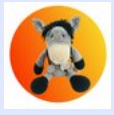
Herman

It's a build, not a belief.



Corn

Alright. I had a job where the whole problem was this.



Herman

You had a job?



Corn

Night shift stock clerk. Regional grocery chain. My job was to walk the aisles and re-shelve everything customers had picked up and put down in the wrong place. That was the entire job. Put the can back where it belonged.



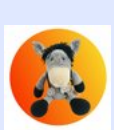
Herman

And the store had a category system.



Corn

It had a beautiful one. Aisle numbers, shelf tags, a whole taxonomy. And it worked. For years, it worked. Then corporate introduced a seasonal aisle.



Herman

Oh no.



Corn

Seasonal overlapped with three other aisles. Cranberry sauce lived in aisle four, except in November, when it lived in seasonal, except the shelf tag in aisle four still said it lived there. So for six months a year I moved the same cans back and forth. Not because anyone was wrong. Because two parts of the system disagreed about where a thing belonged, and neither of them was going to blink.



Herman

And the taxonomy was owned by someone who never shelved anything.



Corn

Someone in an office. The manager would get a memo every quarter and change the layout. We'd just adapt. Nobody asked us. And here's the part I actually came out to say. The best taxonomy I ever saw wasn't the official one. The night crew built our own on a whiteboard in the break room. It wasn't official. It didn't match corporate's aisle numbers at all. But it matched how we actually thought about the products. Where things actually were, in practice, at two in the morning.



Herman

Did corporate ever see it?



Corn

No. Never. We used it anyway.



Herman

Huh.



Corn

I think about that reading Daniel's prompt. Because he's not building the office taxonomy. He's building the whiteboard. He's the one who has to live with the categories, so his ownership isn't a nice-to-have. It's the whole reason the thing will work. The model can't



Corn (continued)

know how you think about your own show.



Herman

That reframes the confidence score for me.



Corn

How so?



Herman

A high-confidence classification means the model is sure. It doesn't mean the category is right. Those are different questions. The whiteboard was right and it wasn't official. The aisle numbers were official and they were wrong half the year.



Corn

Confidence tells you when to look, not when to trust.



Herman

It tells you where the model is unsure. It says nothing about whether your taxonomy matches how listeners actually navigate.



Corn

Which is the thing you can only learn by watching people use it. Alright. I have to go. My sister's expecting me and she's been expecting me for a while.



Herman

How long?



Corn

Long enough that I've stopped counting. She's very patient about it. She mentions it every time.



Herman

Go. I'll finish the thought.



Corn

There's a detail I wanted to get in and it doesn't fit the arc, so I'll put it here. The validation-layer pattern, the one that turns checking everything into checking exceptions, is the thing that makes this sustainable. Without it, Daniel's periodic review is a chore he'll skip. With it, review is twenty minutes and a queue. That's the difference between a taxonomy that stays alive and one that rots.



Herman

The versioned history is the quiet gift. Every taxonomy revision is a record of what you thought the show was about at that moment. For a show built on curiosity, that's not a technical artifact. It's an autobiography.



Corn

Which raises the question I keep circling. If the best taxonomy is the one that matches how people actually think, how do you know when yours is working? Is it when the model classifies with high confidence? Or is it when a listener finds the channel they wanted without ever thinking about the categories at all?



Herman

The second one. And you can't measure it from inside the classifier.



Corn

No. You measure it from the feed. Alright, that's the show. Hilbert Flumingtop produces it, and he's been at the desk the whole time.



Herman

If you're building something like this, a serverless pipeline with a human in the loop on classification, we'd like to hear how you're handling the propagation problem. Leave us a review, or email us at show at my weird prompts dot com.



Corn

This has been My Weird Prompts.



Herman

We'll be back soon.