Episode Book

EPISODE #5697

Decision Models That Return Probabilities, Not Text

PUBLISHED	3 October 2026 at 10:01 UTC
DURATION	27:29
LISTEN	myweirdprompts.com/episode/decision-support-models-probabilities/
HOSTS	Corn, Herman
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:39 UTC

SYNOPSIS

Every launch post for decision support models opens the same way — new class of model, returns probabilities instead of text, benchmark table sorted by accuracy. But accuracy isn't the number your routing system consumes. This episode digs into what these models actually are, why binary yes/no decisions were awkward with ordinary LLMs and JSON constraints, and how they slot into moderation, automations, and agentic systems. Liquid AI's d1 gets a close look, along with the calibration problem that structured outputs never solved.

TRANSCRIPT



Corn

Every article about decision support models opens the same way. There's a new class of model, it returns probabilities instead of text, and here's why that's going to change everything about content moderation. Then it closes with a benchmark table sorted by accuracy.



Herman

And the benchmark table is the part that will get you in trouble.



Corn

Right, because accuracy is not the number the routing system actually consumes. Daniel sent in something on exactly this. He's been watching this category come up, the decision support models, Liquid AI's d1 being the example he named, and his framing is sharp. He says these are built for moderation workflows, and also for automations where you've traditionally had to awkwardly force a yes or no decision through a JSON output structure. His read is that the novelty is that they're explicitly built for that job. Not adapted to it. Built for it.



Herman

That's the correct read, actually.



Corn

So the questions he wants us to get into. What these models actually do. What makes d1 a leading example. Why binary decisions were awkward with ordinary LLMs and JSON constraints in particular. How they slot into moderation, into automations, into agentic systems, into workflows generally. And where this whole class is heading as it matures. That's the shape of it.



Herman

Good. Because the answers to those are more interesting than the launch posts.



Corn

Let's start with what these things actually are. The name is doing a lot of the work and not all of it is honest.



Herman

A decision support model is a language model that does not generate text. That's the whole thing. You hand it a state, which is a ticket, an email, a support thread, a JSON blob, whatever you've got, plus a set of typed questions. It returns calibrated probabilities over a fixed answer set in a single forward pass. Zero output tokens.



Corn

Meaning what, it doesn't write "yes"?



Herman

It doesn't write anything. The answer is read directly out of the model's internal state at a designated position, and then it's restricted to the option set you declared in the request. There's no decoding loop. There's no parser. There's no retry on a malformed response, because nothing can be malformed. The output space is the type. There is no string to get wrong.



Corn

So the entire category of bug where the model returns "yes." with a period and breaks your parser...



Herman

Structurally impossible. That's the pitch, anyway.



Corn

Three primitives show up across every vendor in this space. Noul, which is a yes/no probability from zero to one. Choice, which is a probability distribution across named options. And Score, which is a probability-weighted position on an ordered rubric. Score is



Corn (continued)

the strange one, because it can land between levels. You'll see a severity of two point nine nine nine five on a four-level rubric. Not a three. Almost a three.



Herman

That's the interesting number, honestly. A binary classifier throws that away.



Corn

Timeline. TypeSafe released Jev on the fifteenth of September. That's the thing that created the category. Then Liquid AI's d1 on the twenty-ninth, which is the highest-profile entrant and the one we'll center on. And the key distinction against structured outputs is calibration, not format. That's the part most coverage blurs.



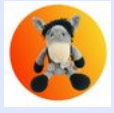
Herman

Right. OpenAI shipped JSON mode back in late twenty twenty-three, then schema-enforced Structured Outputs in August twenty twenty-four. Anthropic added strict structured outputs in twenty twenty-five. All of those guarantee the response parses. None of them guarantee the probability is worth anything.



Corn

So the format problem was solved two years ago.



Herman

Solved completely. Which is why "it returns JSON that validates" is not the story here.



Corn

Then walk me through the mechanism properly, because "non-autoregressive" is the load-bearing word and I want to know what's actually happening.



Herman

A chat model runs an autoregressive loop. It predicts a token, appends it, predicts the next one conditioned on everything so far, appends that, and it keeps going until it emits a stop token. Every step is a full forward pass through the network. That's why a hundred-token answer costs a hundred passes' worth of latency and money, and why the answer can wander, because each token is a fresh decision conditioned on a growing context.



Corn

And the decision model skips the loop entirely.



Herman

It skips the loop entirely. You give it the state and the question. One forward pass. At one designated position in the sequence, you read the logits, you mask them down to just the options the caller declared, and you normalize. What comes out is a probability distribution over your enum. The model never chooses a token. It never writes anything down.



Corn

So the "cannot hallucinate" claim...



Herman

True about types. A decision model cannot invent a category you didn't declare, because the categories are the output space. There's no place for a made-up label to come from.



Corn

And false about content.



Herman

We'll get there, because that's the sharpest thing anyone's written about this whole class.



Corn

Now, why was binary yes/no awkward before? Because I've been forcing models into yes/no answers for years and it mostly worked.



Herman

Mostly is doing the work there. Rohit Raj put it well in September. A chat model that classifies is a text generator you have coerced into behaving, and the coercion leaks. It hallucinates a category that isn't in your enum. It returns "yes" with a trailing period, or "Yes" with a capital Y, or a full sentence explaining that yes, this does appear to violate the policy, and now your strict equality check fails and the whole pipeline stalls.



Corn

The trailing period one has burned me.



Herman

Everyone has been burned by the trailing period. But the format problem is the shallow one. The deeper issue is that a chat model is bad at producing a probability. And there are only two real ways to get one out of it.



Corn

Ask it, or read the logprobs.



Herman

Ask it, and you get a verbalised confidence. "I'm about eighty-five percent confident this is spam." And that number clusters. It clusters at zero point eight five, zero point nine, zero point nine five, regardless of the actual case. A model asked to self-report confidence will produce a narrow band of plausible-sounding numbers whether the case is obvious or ambiguous.



Corn

And reading logprobs is the honest version, right? That's an actual internal quantity.



Herman

It is a real internal quantity. And it's still not trustworthy for this purpose, because RLHF sharpens the distribution. When you train a model on human preferences for confident-sounding answers, you flatten the probability mass onto the chosen token. One test set found thirty-seven of forty cases had over ninety-nine percent of the probability mass sitting on the chosen label. The model isn't ninety-nine percent sure. It's been trained to sound like it is.



Corn

So every case looks like a slam dunk.



Herman

Every case looks like a slam dunk, which is useless for routing, because routing is exactly the job of telling the slam dunks from the not-slam-dunks. And if you're on Anthropic's API, logprobs aren't exposed at all, so that route is simply closed.



Corn

There's a deeper layer to this though. The reason verbalised confidence is so badly calibrated is that it learned confidence from human text, and humans are a disaster at it.



Herman

A legendary disaster. Sherman Kent did this work at the CIA in nineteen sixty-four. He asked analysts what phrases like "serious possibility" meant as a probability. The answers spanned from twenty percent to eighty percent. Same phrase, same building, same decade, a sixty-point spread.



Corn

And that got replicated.



Herman

Replicated in twenty fifteen with forty-six people, and again this year with ninety-nine, and you get the same pattern. A ten to twenty point interquartile spread on the same phrase. When a model learns to say "serious possibility" from human text, it is learning a convention that humans never actually agreed on.



Corn

So the verbalised confidence isn't just noisy. It's inheriting a calibration error that predates computers.



Herman

That's the whole argument for training calibration directly instead of hoping it emerges from next-token prediction. If your training signal rewards being right at the confidence you claimed, rather than rewarding sounding right, you get a different animal.



Corn

Which brings us to d1. Liquid AI, released the twenty-ninth of September. First model to overtake Jev on Hugging Face's Jev Decision Index, and that's a self-reported lead, which we'll come back to.



Herman

Self-reported, yes. Their API is an endpoint that posts to a decisions path, model id d1 colon free, API-only. No public weights, no GGUF, nothing on Hugging Face from them. Hosted proprietary.



Corn

And it reports output tokens as zero in every response.



Herman

Because there are no output tokens. That's not a marketing claim, it's a structural fact about the architecture.



Corn

The thing I find useful is that one request can mix question types against the same state. So a support system can ask for the intent label, an urgency score, and a probability that this is a bug, all in one round trip, against one pass over the text.



Herman

One pass. That matters when you're doing this at volume, because the alternative is three separate calls to a chat model, each with its own latency, each with its own chance of drifting because you changed the preamble.



Corn

What did Liquid claim over their earlier decision models?



Herman

Four things. Higher multilingual scores. Greater prompt-injection resistance in the state field, which is interesting because the state is where untrusted text lives. Better handling of long inputs. And faster structured decisions.



Corn

The injection resistance one is the one I'd want to see independently tested.



Herman

Same. And it's on Vercel's AI Gateway now at four cents per million input tokens, zero for output, with a sixty-six thousand token context window.



Corn

And they shipped a cookbook demo called Road Decider.



Herman

A pixel-art driving game where d1 picks left, center, or right every frame, with a confidence score attached.



Corn

It's a driving game where the car is being steered by a classifier.



Herman

That is exactly what it is, and I think it's a better illustration of the class than any benchmark table, because you can see the confidence number in real time and you can see what happens when it's wrong.



Corn

So that's the mechanism. Now let's talk about where these things actually get deployed, and the use case that made the category in the first place, which is moderation.



Herman

Moderation is the canonical case, and the reason is simple. A confidence number you cannot trust means either over-removal or under-enforcement, and both of those are visible. You either delete things you shouldn't have, and people notice, or you leave things up you shouldn't have, and people notice harder.



Corn

The recommended pattern here is not "ask for a label." Walk me through it.



Herman

You use a Score question for the severity tiers. Benign, questionable, harmful, severe. And then separate Noul questions for the specific policies you care about. Does this target a real person. Is this commercial spam. Those come back as independent probabilities, not as a single tangled label.



Corn

And the thresholds live in code.



Herman

The thresholds live in code. Which means tuning your moderation policy is a config change, not a model change. You move the line from zero point seven to zero point six, redeploy the config, done. You're not re-prompting, you're not re-fine-tuning, you're not filing a ticket with a vendor.



Corn

That's the part that would have saved me months. Because with a prompt-based classifier, tuning the policy meant changing the prompt, which changed the behavior on cases you weren't trying to touch.



Herman

And you'd never know which ones until they showed up in the queue.



Corn

So an actual worked example. There's one measured in mid-September on a rude-but-not-abusive post.



Herman

Severity zero point three two out of three. Benign sixty-nine percent, questionable twenty-nine percent, harmful two percent. Targets a real person, zero point zero five. Is spam, zero point zero two.



Corn

So the honest description of that post is "this is in the band where a binary classifier is useless and a distribution is not."



Herman

That's the sentence. A binary classifier has to call that one way or the other, and either call is defensible and either call is wrong some of the time. The distribution says: this is mostly benign with a real minority of questionable in it, and the flags for the two hard policies are both near zero. That's actionable in a way "harmful: false" is not.



Corn

And the price.



Herman

Zero point zero zero zero zero one seven six four per call at four hundred and twenty tokens. Roughly seventeen dollars and sixty-four cents per million posts.



Corn

So the cost of moderating a million posts is less than a decent lunch.



Herman

Which changes the argument entirely, because at that price the question stops being "can we afford to check everything" and becomes "what do we do with the numbers."



Corn

The three-lane flow. Clear passes publish. Clear violations get removed and logged. Everything in between goes to human review with the distribution attached.



Herman

And the width of that middle lane is a dial you turn. That's the design decision. You're not choosing a classifier, you're choosing how wide the review band is.



Corn

Now the caveats, because everyone skips these and they matter.



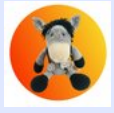
Herman

Most harmful material is image and video, and these models are text-only. They moderate captions, comments, and transcripts. If your problem is the video itself, this class does nothing for you.



Corn

Users probe filters with "it's just satire" framing.



Herman

Which is hard, and a severity distribution handles it better than a yes/no, but it doesn't make it easy. And every removal needs a written reason, and the model can't supply one. You get the number, you write the sentence. Non-English accuracy is also lower, though Liquid is claiming improvement there specifically.



Corn

Moderation is the canonical case, but the more interesting question is what happens when you put one of these inside an agent loop.



Herman

That's where it gets architectural. Because an agent generates a lot of its own sub-questions during a run. Should I pursue this. Is this result stale. Does this tool call need approval. Is this output worth keeping in context.



Corn

And a chat model answering those questions is expensive and slow, and you're burning reasoning tokens on yes/no decisions.



Herman

So you insert the decision model as a tool call. It answers the sub-question in one pass for a fraction of the cost, and the agent routes on it. One commenter on Hacker News framed it as the decision model selecting which questions are worth pursuing, which is a good way to put it.



Corn

And there's shipped tooling doing exactly this.



Herman

Two that I'd point at. One is a Claude Code plugin that scores every tool call and result and drops the ones that have gone stale, seven thousand one hundred stars. The other trims long shell output before the model ever sees it, a hundred and fifty-two stars. Both are just decision models hooked into an agent's context management.



Corn

So the practical use cases stack up pretty fast. Ticket routing. Email routing. Safety flags. Tool-call approval gates. Rubric-based evaluations. Model routing and inference cascades. Fraud and risk scoring. Reranking retrieval results. LLM-as-judge checks.



Herman

And Laya's preset question sets show the agent pattern nicely. There's a guard set for input filtering, jailbreak, prompt injection, sensitive data, harm severity. There's a router set for deciding which model tier handles a request, based on difficulty, domain, whether tools are needed, whether it's sensitive. And a triage set for intent and urgency.



Corn

The guard set is the one I'd want in production. A jailbreak probability before the expensive model sees the prompt.



Herman

And that's an input guardrail that costs four cents per million tokens and adds one forward pass. That's a different calculus from running a full model to check.



Corn

Now the economic argument, which I think is the most underrated thing about this whole class.



Herman

In a cascade where ninety-five percent of cases get auto-handled, four percent escalate to a reasoning model, and one percent go to a human, the cost of the whole workflow is set almost entirely by those last five percent. The cheap tier is nearly free. The expensive tier is where the money goes.



Corn

Which means the value of a good probability is not in the ninety-five. It's in knowing which five to send up.



Herman

And that's why calibration is the buying criterion. Accuracy tells you how often you're right overall. Calibration tells you whether the eighty-percent-confident cases are actually right eighty percent of the time. If they are, you can route on the number. If they aren't, your



Herman (continued)

threshold is a guess that looks like a policy.



Corn

There's a line I keep thinking about from the open-weights comparison. Most decision-model use is routing, so most of the time the calibration column is the buying criterion.



Herman

That's the whole argument in one sentence, and it cuts against how literally every leaderboard is sorted.



Corn

Which brings us to the landscape, because it got crowded in about two and a half weeks.



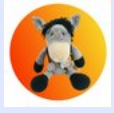
Herman

JevBench as of yesterday ranks a hundred and six systems. Top of the board is a frozen Gemma-based model at seventy-three point seven. Then a twelve-billion at seventy-three point two three. Jev itself at seventy-two point one three. The spread at the top is under two points.



Corn

Two points across the top four.



Herman

Two points. Which tells you the accuracy race is essentially over at the frontier, and the interesting competition has moved elsewhere.



Corn

The open-weight tier is the part that surprised me, given the category is eighteen days old.



Herman

Laya is a four hundred and twenty-one million parameter model under Apache two, and it runs at about thirty-three milliseconds per question on a T4, and between a hundred and ninety and four hundred and sixty milliseconds on plain CPU. There's a multilingual variant covering over a hundred languages.



Corn

CPU. No GPU required.



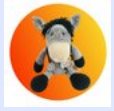
Herman

That's the thing. You can run this on a laptop, or on a small VM, for basically the cost of the electricity.



Corn

And Kev, which is the accuracy leader in the open tier.



Herman

Four point six thousand stars, Apache two, Qwen-based. Kev at nine billion scores zero point eight five two accuracy against Jev's zero point eight five seven. Essentially tied on accuracy.



Corn

And then open-alternative-jev, which is the calibration leader.



Herman

ECE of zero point zero two zero. That's the best-calibrated model in the comparison. And here's the split that matters: sort by accuracy, Kev wins. Sort by calibration, open-alternative-jev wins. They're not the same model.



Corn

So the model you want depends entirely on what you're doing with the number.



Herman

If you're reporting accuracy to somebody, Kev. If you're setting a threshold and acting on it, the calibrated one, even though it loses the accuracy race. And for reference, Laya's calibration error was zero point four six six out of the box, and dropped to zero point zero eight one after a temperature fit. That's a five point seven times improvement from fitting one scalar.



Corn

For people who haven't done this. ECE under about zero point zero five is usable for threshold routing. Over about zero point one five, your thresholds are fiction. And fifty to three hundred labelled examples plus one fitted temperature scalar can cut calibration error by up to seventy-four percent.



Herman

Three hundred examples. That's a weekend of work, not a research project.



Corn

And self-hosting crossover. About two million decisions a month before a hosted API makes more sense than running your own GPU. Except CPU-only Laya moves that to nearly any volume.



Herman

Because there's no GPU to amortize. You're just running it.



Corn

There's a counter-argument here that deserves airtime before we wrap up.



Herman

Red Hat published a piece yesterday arguing decision models don't beat LLM-as-a-judge or traditional classifiers. And the top comment on the discussion thread was essentially "duh, this is not news."



Corn

Which is not an unreasonable reaction.



Herman

It's not. If you've got a fine-tuned BERT that does your specific classification task at ninety-two percent, a general decision model at seventy-three is not an upgrade. And that's a real answer for a lot of teams.



Corn

What's the counter-position?



Herman

That decision models are the first option that is general, fast, and cheap simultaneously. You used to pick two. A traditional classifier is fast and cheap but only does the one thing you trained it on. A chat model is general but slow and expensive. This is the first thing that's all three at once, and the tradeoff is that it's not the best at any one of them.



Corn

The sharpest push, which I think is the thing to leave people with. A decision model that cannot hallucinate types can still be confidently wrong about content. And because the output is a clean float, it looks more trustworthy than a chat model's hedged paragraph. The type safety is real and the epistemic safety is not.



Herman

That's the sentence. Because a chat model hedges. It says "this appears to violate the policy, though the context is ambiguous." You read that and you're on guard. A decision model returns zero point nine one and you just… believe it. The format is doing persuasive work the content hasn't earned.



Corn

Simon Willison flagged some of this. Jev does poorly with numbers, dates, and adversarial content. No interpretability. And an unexplained geographic bias, where Cupertino rated favourably and East Palo Alto unfavourably on the same kind of content.



Herman

Same content, different town, different answer. And you cannot ask the model why, because there is no why. There's a float.



Corn

The clean number is the problem, not the solution.



Herman

The clean number is the problem. It moves the ambiguity somewhere you can't see it.



Corn

Herman.



Herman

Mm.



Corn

The moderation workflow you described. Three lanes. Clear pass, clear fail, and the middle band that goes to a human.



Herman

Yes.



Corn

The middle band had a name in a previous life. It was called the amber band, and it was about a third of the board on a good night.



Herman

Where's that from?



Corn

Dispatch board. Regional courier outfit, night shift. There was a confidence field, color-coded. Green, amber, red. The amber band was where every bad call lived.



Herman

Every bad call.



Corn

Green you shipped, red you held. Amber you called the customer, because the number had told you nothing.



Herman

They routed around it.



Corn

They routed around it. And management kept trying to shrink the band by moving the thresholds, not by improving the signal. Which is exactly the thing you said about thresholds living in code. You can turn that dial, but turning it doesn't make the underlying number better.



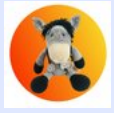
Herman

The dispatchers figured out that the model was wrong in a specific place, and the specific place was the band they had to act in.



Corn

The accuracy overall was fine. Eighty-something percent, probably. Nobody cared. The fifteen percent it got wrong was concentrated exactly where a decision had to be made.



Herman

That's the whole calibration argument in a courier depot.



Corn

And the part nobody writes down is that the people setting the thresholds were not the people eating the consequences of a wrong threshold. The dispatchers were. So they learned to distrust the number entirely.



Herman

Which is a human routing around a miscalibrated model, in the most literal sense.



Corn

That's a thing I'm going to be thinking about every time somebody shows me a leaderboard. The board was accurate. The board was useless.



Herman

It wasn't useless. It was useless in the band that mattered.



Corn

Say the difference.



Herman

The board told them where the easy calls were. It just didn't help with the hard ones, and the hard ones were the job.



Corn

Alright. Let's land this. The category is eighteen days old.



Herman

Jev shipped the fifteenth of September. d1 on the twenty-ninth. And the question somebody asked on the first of October is the one that should bother everyone: how did so many people build decision models within days or weeks of Jev coming out? Was this in the works for a while, or is it easy to copy?



Corn

The answer is both, and neither is comforting. The interface is trivially copyable. It's a forward pass and a masked distribution over a declared option set. Anybody with a base model and a labelling pipeline can build the shape. What isn't published is the calibration recipe.



Herman

TypeSafe's training method, Reinforcement Learning for Calibrated Decisions, has no paper. No reward function, no base model, no parameter count, as of three weeks in. Every third-party explainer paraphrases the same three sentences from the docs. And the acronym collides with an unrelated twenty twenty-three method, so searching for it returns the wrong paper entirely.



Corn

Which means the thing that actually makes these models work is the thing nobody has shown.



Herman

The type safety is real. The epistemic safety is not. And the amber band is still there. It's just been moved into code.



Corn

It's the same three lanes, and the middle one is still a third of the board on a good night. The difference is now the number has decimal places.



Herman

Which makes it look like it knows.



Corn

That's the show. Thanks to Hilbert Flumingtop for producing. This has been My Weird Prompts, the human-AI collaboration podcast.



Herman

If you got something out of this, a review helps other people find it.



Corn

We'll be back soon.