Episode Book

EPISODE #5467

Small Models, Big Schemas: When JSON Constraints Backfire

PUBLISHED	22 September 2026 at 15:16 UTC
DURATION	20:21
LISTEN	myweirdprompts.com/episode/small-models-structured-output/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:36 UTC

SYNOPSIS

Everyone building agents reaches for the biggest model and wraps every endpoint as a tool. But there's a quieter pattern: tiny text-to-text models doing one narrow structured job. Daniel wrote in asking whether small models doing predictable structured output inside MCP pipelines actually ship in production, or whether everyone just lets a general-purpose model emit tool calls. The answer runs through Todoist's MCP servers, a 7B structured-object model from Amazon Science, and a May 2026 study called The Constraint Tax, which found that hard answer-only schemas pushed schema validity to 100% while answer accuracy fell from 19.7% to 11%. Then FlowMCP flips the whole premise: maybe the win isn't a smaller model, it's a smaller surface.

TRANSCRIPT



Corn

Most teams building agents today reach for the biggest model they can afford, wrap every endpoint they own as a tool, and then act surprised when the thing wobbles.



Herman

And the fix they reach for is usually a better model.



Corn

Which is expensive, slow, and often not the actual problem. Daniel's been poking at the other end of this. He wrote in about the unglamorous long tail on Hugging Face, the text-to-text models that aren't large language models at all, small enough to run on a CPU. His own example is a classifier he wants to build that finds Hebrew words written in Latin characters and converts them to Hebrew Unicode so a text-to-speech engine reads them in the right language.



Herman

Which is a very specific job.



Corn

Very. And he wants to stretch that idea out. A small model that takes a freeform to-do list and structures it for an MCP server to push into a task manager. A model that turns dictated meeting plans into iCal events. A model that parses a dictation into email fields, subject line, CC, body. Then the real question underneath all of it. Is this pattern actually used in production, small models doing predictable structured output inside agentic pipelines with MCP? Or is everyone just doing the classic thing, where a general-purpose model reads the tool definitions and emits the tool calls itself? And he wants the pros and cons of classifiers and small models against that classic approach for this workload.



Corn

Please.



Herman

Small language models, conventionally, are somewhere between half a billion and eight billion parameters. That's the range where you can run the thing on a single GPU, a laptop, sometimes a phone CPU. Some surveys stretch the ceiling to twelve billion, occasionally twenty. The enterprise sweet spot people keep landing on is one to seven billion. The frontier hosted models are hundreds of billions of parameters, and that's the comparison everyone makes by default.



Corn

And a classifier is a subtype.



Herman

A classifier assigns labels. It doesn't generate open-ended text, it picks from a set. Structured extraction and routing sit right next to it and usually get lumped in. The tasks small models actually win on are focused classification, structured extraction, routing, and short formatted output. That's the list, over and over.



Corn

So the question isn't which model is smarter.



Herman

It's which model is correct enough for one specific task at a fraction of the cost and the latency. That's the entire frame. And the episode has an arc. The use cases are real. The classic approach dominates what's actually shipped. The small-model-for-structured-output pattern is real in research and in narrow deployments. And then there's a finding that complicates the whole premise, which we'll get to.



Corn

Start with the shipped reality, because Daniel's use cases aren't hypothetical.



Herman

Todoist MCP servers are everywhere and they're production-grade. Doist ships their own. There's one from greirson with nineteen tools. There's learhy's, which literally advertises itself as production-grade. Several of them say, right in the README, quick add tasks, use natural language to create tasks. And Todoist ships an official Claude Code integration through a CLI or an MCP server.



Corn

So the pattern in those servers is what, exactly?



Herman

Exactly the classic approach Daniel described. The general-purpose model reads the tool definitions, parses the user's text, and emits the tool calls. That's it. That's the whole architecture.



Corn

Here's the part I find funny. Todoist already has a natural-language quick-add parser.



Herman

It does.



Corn

So the freeform-to-structured conversion, in a lot of these flows, isn't being done by a model at all. It's deterministic app logic that's been sitting there for years.



Herman

Which is the first hint of where this episode is going. The most reliable component in the pipeline is the boring one.



Corn

But narrow structured-output models are a real pattern, not just a wish.



Herman

Amazon Science published work on what they call a structured-object language model. Seven billion parameters, trained for one thing only, emitting objects in a specific schema. Self-supervised denoising to train it, and a custom decoder they built called confidence-aware substructure beam search. It matched or exceeded state-of-the-art LLMs on output accuracy at an order of magnitude better cost efficiency. The decoder alone improved recall by sixteen point seven percent at fixed ninety percent precision.



Corn

Trained to emit objects in one schema. That's a very narrow animal.



Herman

It's a machine that produces one shape of thing. And there's a survey from last October, Sharma and Mehta, on small language models for agentic systems. Their line is that SLMs are sufficient and often superior for agentic workloads where the objective is schema- and API-constrained accuracy rather than open-ended generation. Guided decoding plus strict JSON Schema, they argue, often lets small models match or surpass large ones on tool use, function calling, and retrieval, at ten to a hundred times lower token cost.



Corn

So on paper, Daniel's instinct is well supported.



Herman

On paper.



Corn

That's a pause.



Herman

Because there's a paper from this May by Jaideep Ray called The Constraint Tax, and it changes the calculus.



Corn

Go.



Herman

Fifteen thousand generations on commodity GPUs. Qwen two point five at half a billion and one point five billion, SmolLM2 at one point seven billion. He ran the same tasks two ways. Freeform, and under a hard answer-only schema. Schema validity went from sixty-one point five percent to one hundred percent. Perfect. Every output parseable.



Corn

And the answers?



Herman

Answer accuracy fell from nineteen point seven percent to eleven percent.



Corn

Hold on.



Herman

And outputs that were valid but wrong went from forty-nine point five percent to eighty-eight point nine percent.



Corn

Say that again. The schema made them worse at the task.



Herman

It made them worse at the task while making them look better on every dashboard. That's the finding. Hard schemas don't just change the wrapper. For small models, they change the answer.



Corn

Why? What's the mechanism?



Herman

The constraint is applied during decoding. The model is choosing the next token, and the schema is pruning the token space at every step. So the model can't lay out a partial thought, can't hold an intermediate value in free text and come back to it. It has to commit to a field, in order, immediately. For a small model, that's a lot of its working memory gone.



Corn

It's doing arithmetic with one hand tied.



Herman

Worse than that. It's doing arithmetic while being graded on handwriting.



Corn

And the most production-like result?



Herman

Calendar tool calls. Prompt-only JSON hit ninety-one point five percent executable accuracy. The same task under a hard tool-call schema hit forty-eight percent. Both one hundred percent schema-valid.



Corn

Forty-three points, gone.



Herman

The errors were semantic, not structural. One example. Asked for a thirty-minute meeting with Leo, the constrained decoder emitted a perfectly valid object with duration in minutes set to one hundred and eighty.



Corn

Three hours with Leo.



Herman

A hundred and two of the hundred and four hard-schema failures were that single field. duration_minutes. Everything else was right.



Corn

And it isn't just the tiny models.



Herman

Qwen two point five at three billion lost fifteen point three answer-accuracy points under hard schema, with wrong-valid outputs up thirty-one point six points. The tax doesn't vanish when you cross the three-billion line.



Corn

So the intuition Daniel's whole idea rests on, that a tight schema makes a small model safe, is measurably wrong.



Herman

That's the twist. And Ray's line is the one to remember. A valid JSON object can still encode the wrong decision, so a dashboard that tracks parse success alone can improve while downstream execution gets worse. Parseability is a transport property, not a task-success metric.



Corn

Which is a sentence every team building an agent should have taped to the monitor.



Herman

And his recommended pattern is reason free, constrain late. Let the model solve the task in freeform, then project the result into the executable object afterward. Delayed constraint reached one hundred percent schema validity while keeping the highest executable accuracy, forty point seven percent against twenty-six point eight for direct answer-only schema.



Corn

So the order of operations is the fix.



Herman

Solve first, shape second. Which is not how most of these pipelines are built. Most of them constrain at the first token because that's the easiest thing to wire up.



Corn

But here's where it gets interesting, because the real lever may not be the model at all.



Herman

This is the part I find exciting. FlowMCP. It's an MCP design pattern, version zero point nine point three, MIT licensed. The idea is that workflows are the tools. Each MCP tool is one deterministic workflow, and the model's only job is picking the flow and filling in two or three parameters.



Corn

The model picks a door and turns a knob.



Herman

And the README says small models, seven to thirty billion, can drive this reliably because there is almost nothing to get wrong. Their benchmark, frozen at the end of July, reports seventy-nine percent task success through a two-flow façade against ten percent on the same thirty-five tools raw, at a tenth of the tokens per attempt.



Corn

Ten percent to seventy-nine.



Herman

Same tools. Same model. Different surface. And the headline result, a seven-billion model through the façade outscored a thirty-five-billion model driving the raw surface. Six hundred and ten runs executed, four hundred and forty analyzed, thirty-three discordant pairs, exact p around two point three times ten to the negative tenth.



Corn

So Daniel's question might be the wrong question.



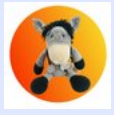
Herman

Maybe the win isn't a small model. Maybe it's a small surface. The FlowMCP README is blunt about it. Most MCP servers wrap an entire platform, every endpoint becomes a tool, the model gets a forty-tool surface, and then everyone blames the model.



Corn

Forty tools is a filing cabinet with no labels.



Herman

It's a filing cabinet where the model has to remember which drawer is which, every single time, from scratch.



Corn

Now, production reality. Replacement or hybrid?



Herman

Hybrid, and the research is consistent on it. Small model default, large model fallback, with a router and a verifier cascade. The survey proposes the metrics you'd actually track. Cost per successful task, schema validity rate, executable call rate, p50 and p95 latency, energy per request.



Corn

Cost per successful task. Not cost per call.



Herman

Because a cheap call that produces a wrong-but-valid object is not cheap. It's a ticket.



Corn

Give me the case study.



Herman

A team integrating Gemma 4 E2B, two point six billion, and Qwen3 at zero point six billion into a production Android game. Five days, two hundred and four commits. And they retreated. They started out having the model generate complete structured puzzles as JSON, and they backed off to a pragmatic architecture where curated word lists provide the words and the model generates only three short hints, with a deterministic fallback.



Corn

They took the structured generation away from the model.



Herman

They took most of it away. The model kept the narrow generative bit and the deterministic code kept everything else. Their five failure categories are worth reading. Output format violations, constraint violations, context quality degradation, latency incompatibility, model selection instability.



Corn

Model selection instability is the one people don't plan for.



Herman

You pick a model, you ship, the provider updates it, and your pipeline's behavior shifts under you. That's a real operational cost of renting.



Corn

And there's a comment from a practitioner on Hacker News in May that describes the pattern in one breath.



Herman

He says he uses a super small one, fine-tuned to do function calling, instead of sending it to a big model and waiting. And that what's great with the smaller ones is you get huge throughput with just vllm and a couple of consumer GPUs. And his workflow is distillation, a big one onto a small one.



Corn

That's the whole recipe. Teacher, student, consumer hardware.



Herman

And the numbers back it. A fine-tuned one point five billion model cut latency eighty-seven percent and cost per request ninety-six percent against a hosted LLM, hitting sub-three-hundred-millisecond targets with a ninety-nine point five percent valid-JSON rate. With one caveat they're honest about. Accuracy drops when the function schemas change, so you keep a fallback for functions the model has never seen.



Herman

It's the whole argument for the hybrid. The small model owns the stable surface. The large model handles the novel.



Corn

And the fine-tuning cost itself?



Herman

A DigitalOcean experiment fine-tuned Qwen two point five at seven billion with LoRA in three point two minutes, for a few cents, and it beat a prompted Llama three point three at seventy billion on quality while answering about fifteen times faster, on two hundred



Herman (continued)

and fifty intent-classification items.



Corn

A few cents.



Herman

The 70B only won on cost at low volume. And even that flips.



Corn

To the flip point.



Herman

A fine-tuned frontier model that feels free at ten thousand calls a month costs around forty thousand dollars a month at ten million plus calls a month. A custom small model on a single GPU runs around two thousand dollars a month. Twenty times apart.



Corn

So should I use a small model is really, is this workload high-volume, low-complexity, and stable enough that owning the model beats renting it.



Herman

That's the question. At low volume, renting wins. At high volume with a stable task shape, owning wins decisively. There's no universal answer, and anyone selling you one is selling you something.



Corn

Now the honest answer to what Daniel actually asked.



Herman

The classic approach dominates in shipped MCP servers today. That's not close. The small-model-for-structured-output pattern is real in research and in narrow fine-tuned deployments, but it is not yet the default in off-the-shelf MCP tooling. There's no standalone product you can buy that's a small classifier for freeform-to-structured MCP ingestion. The closest real artifacts are Amazon's structured-object model, FlowMCP's workflow-as-tool pattern, and the pile of Todoist servers doing the classic thing.



Corn

And his Hebrew classifier?



Herman

Nothing. No model, no paper. Searches for neural transliteration of Hebrew from Latin script came back empty. It's a genuine gap.



Corn

That's a useful finding, actually.



Herman

It's the most useful finding in the whole thing. Not every narrow task has a model waiting for you. Some of them you build, and this one looks like a small fine-tuned classifier or a sequence-to-sequence model you train yourself.



Corn

So the long tail has holes in it.



Herman

The long tail has holes in it, and the holes are where the interesting work is.



Hilbert

Four hundred and twelve dollars.



Corn

What's four hundred and twelve dollars?



Hilbert

That's what the dispatch terminal cost, the one I keyed into. Regional parcel company, mid eighties. Handwritten driver notes came in on a clipboard and I typed the corrections into the legacy system. Address changes, mostly. Wrong street number, apartment added, that kind of thing.



Herman

That's the job. That's exactly the job.



Hilbert

It's freeform to structured, done by a person. And I was good at it. Best on the floor for about a year.



Corn

What made you good at it?



Hilbert

I read the whole note first. All of it, top to bottom, before I touched a field. Then I knew what the driver meant, and I filled in the fields to match. Supervisor didn't like it. Said it looked like I was sitting there doing nothing. So he made me fill the fields as I read. Start with the postal code, then the street, then the note.



Herman

And your error rate went up.



Hilbert

It went up. I started putting the right-looking thing in the wrong field. Correct postal code, correct street, correct house number, wrong town. The system never flagged them. Every field validated. The customer called three days later.



Corn

Wrong but valid.



Hilbert

That's what the paper calls it. Wrong-valid-schema. Took fifteen thousand generations to say what my supervisor should have known. The bad ones were never the messy ones. The messy ones I caught. The bad ones looked perfect.



Herman

So reason free, constrain late isn't new.



Hilbert

It's how anybody who was any good at that job did it. Read the whole thing, understand it, then write it down in the boxes. The boxes come last. If you put the boxes first, you're not reading the note anymore, you're filling in boxes.



Corn

And nobody told you it wasn't you.



Hilbert

Nobody told me it wasn't me. I figured I'd gotten worse. Anyway. Something outside needs moving before it gets dark.



Corn

The cutting-room floor. One thing from the research that didn't fit.



Herman

The delayed-constraint numbers have a detail I keep chewing on. Reason free, constrain late gets you to a hundred percent schema validity with the best executable accuracy in the study, forty point seven percent. But forty point seven percent is still the best number in that table. Even the winning pattern in a sub-two-billion-parameter regime is failing most of the time on that task.



Corn

The honest read is that the small model isn't the hero of that paper.



Herman

The small model is the constraint. The paper is telling you where the floor is.



Corn

Which leaves a question hanging. If the real lever is tool-surface design, then the MCP ecosystem's habit of wrapping every endpoint as a tool might be the actual problem, not the models. And if reason free, constrain late is the right pattern, how do you implement it in a live pipeline without adding a second pass and the latency that comes with it? And if the Hebrew classifier doesn't exist, how many other narrow tasks are sitting out there waiting for someone to build the model?



Herman

The economics point makes the rest of it a volume question. At ten thousand calls a month, rent. At ten million, own. And MCP is at ninety-seven million monthly SDK downloads and thirteen thousand public servers, so the volume is arriving whether the tooling is ready or not.



Corn

Which is the warning. The constraint tax says your reliability dashboard can improve while your actual task success gets worse. That's a bad way to find out.



Herman

Thanks to Hilbert Flumingtop for producing.



Corn

This has been My Weird Prompts. If you enjoyed this deep dive into the unglamorous long tail, subscribe at my weird prompts dot com or in your podcast app.



Herman

We'll be back soon with another weird prompt.



Corn

See you tomorrow.