# Episode Book

EPISODE #6056

# Where Does Speaker Structure Belong in AI Podcast Scripts?

| | |
|---|---|
| PUBLISHED | 17 October 2026 at 14:23 UTC |
| DURATION | 25:29 |
| LISTEN | myweirdprompts.com/episode/speaker-attribution-structured-output-pipelines-05763/ |
| HOSTS | Corn, Herman, Larry, Hilbert |
| SCRIPT MODEL | deepseek-flash |
| VOICES | chatterbox-regular (chatterbox-local) |
| PIPELINE | V5.10 |
| SOURCES | 13 cited by the research (listed at the end) |
| BOOK GENERATED | 19 October 2026 at 17:43 UTC |

## SYNOPSIS

Every multi-speaker AI pipeline runs on a contract with no enforcement layer — and when it breaks, the failure gets blamed on the model. This episode digs into a documented bug report from an open-source podcast generator: six segments, three speakers, forty-two dialogue lines, and a model that returned the prompt's own JSON skeleton verbatim. Then it turns to the obvious fix — strict structured output — and what two research papers actually found about the cost of asking a model to respond in JSON. The answer isn't where most people assume: the damage happens at the prompt, not the decoder. Along the way: why high-temperature sampling doesn't restore diversity under a template, why capacity matters more than format for strong models, and why "thinking first" helps math but does nothing for prose.

# TRANSCRIPT



**Corn**

Every episode of this show starts as one long block of text. Lines with our names on them, and somewhere downstream a parser splits it so my voice reads my lines and Herman's voice reads his.



**Herman**

And nothing forces the model to get that right. Nothing at all.



**Corn**

That's Daniel's question. He's been watching this pipeline do exactly what it does, which is mostly work and occasionally hand my line to Herman. He lays out the whole thing: a model bolds a name, writes "HERMAN (laughing):", and the line quietly ends up in the wrong mouth. The obvious fix is strict JSON, a neat list of speaker and line pairs from one of the structured-output modes every API sells now. Except he's heard that forcing JSON makes creative writing worse, and it doesn't stop the model from handing a line to the wrong character anyway.



**Herman**

Two claims, both worth testing.



**Corn**

And then the real one underneath. For long, multi-speaker creative text, where does structure belong. In the model's output, in a validator after the fact, or trained into the model itself.



**Herman**

Let's start with why the obvious thing keeps breaking, because it breaks for a better reason than people think.



**Herman**

The label-and-parse design is a contract with no enforcement layer. You and the model have agreed on a format, and nothing in the system can hold either side to it. I tell the model "write Corn colon, then the line," and the model writes it nine times out of ten, and the tenth time it writes "Corn:" in bold, or puts a stage direction in parentheses, and now you have a token that isn't quite the token your parser is looking for. Either validation aborts on it, or the parser is lenient and glues the line to the wrong voice.



**Corn**

And bolding a name feels like such a small thing.



**Herman**

Because it is. That's the point. It's a formatting deviation, not a mistake about who's speaking. The model knows perfectly well it's writing my line. It just decorated the label.



**Corn**

So the parser is the one that doesn't know.



**Herman**

The parser has one job and it's brittle by design. It's looking for an exact string. Anything that isn't the exact string is either an error or, worse, a silent success in the wrong place.



**Corn**

There's a case I want to get into, because it's the best-documented version of this I've seen.



**Herman**

The open-notebook one.



**Corn**

Podcast generator, open source, and back in August somebody filed a bug report on it. Six segments, Hebrew episode, forty-two dialogue lines across three speakers. And the failure is beautiful, honestly. The prompt hands the model a fill-in JSON skeleton, with a speaker field and a placeholder value in brackets, and then a bare ellipsis as the continuation.



**Herman**

And the model returned the skeleton verbatim.



**Corn**

Word for word.



**Herman**

The reporter's line is the one worth keeping. "That completion is not a hallucination. It is the prompt's own example, returned verbatim."



**Corn**

Say that again. Because that reframes the lot.



**Herman**

It's a contract mismatch. An ellipsis inside a JSON example is indistinguishable from an ellipsis as a value. The model looked at a template full of blanks, and blanks are for filling, and it filled them with the blanks that were already there. Then validation threw a hard error: invalid speaker name, dot dot dot. Must be one of three names.



**Corn**

Which names.



**Herman**

Marcus Thompson, Elena Vasquez, Johny Bing. None of whom is a placeholder.



**Corn**

So the model was given a form and it handed the form back.



**Herman**

And here's the part that I find instructive, because it's not a model failure at all. The schema the model saw carried no speaker list. The format instructions were rendered from an unvalidated transcript model. The runtime parsing used a validated one, with a whitelist of allowed speakers. The contract shown was not the contract enforced.



**Corn**

The allowed names reached the model only as prose.



**Herman**

Somewhere in the middle of a paragraph of instructions. Not as an enum. Not as a constraint. As a sentence.



**Corn**

That's the whole bug right there.



**Herman**

And the prompt also contradicted itself about code fences. One line opened a json block. Another line, further down, said do not wrap the JSON in json code blocks. And contradictions push a model toward pattern matching over instruction following. When the instructions argue with each other, the model picks the pattern it likes best.



**Corn**

So it picked the pattern where you copy what you were given.



**Herman**

It picked the pattern where the answer is already on the page. And then the failure was late and misattributed, which is the piece that would drive me up the wall. Segment two of six aborted after several minutes and a full transcript's worth of tokens. The error hint named the wrong cause. Segment one had already succeeded and got thrown away with it.



**Corn**

You paid for all of it twice.



**Herman**

You paid once and threw it away. Which is worse.



**Corn**

This is a known hazard downstream, isn't it. The whole tooling ecosystem around podcast generation treats speaker labels as a place where things go missing.



**Herman**

Jellypod's documentation describes resolving bracketed speaker labels to saved characters, and then warns that explicit assignments override name matching. Which is a polite way of saying if your explicit assignment is wrong, the name match never gets a chance to save you.



**Corn**

And there's a scraper project with a decision record about the mirror image of this.



**Herman**

ADR-137. Rotating-host shows. Planet Money is the example they use. Those shows state no fixed hosts, so the show's own hosts get labeled as guests. And a stated guest who never introduces himself never gets bound to a voice at all. All voices unresolved.



**Corn**

So a show with no fixed host breaks the assumption that a host exists.



**Herman**

And a show with three hosts breaks the assumption that the label is one of two things. It's the same assumption failing in both directions.



**Corn**

Alright. So label and parse fails because it's a contract with no enforcement. Obvious fix, make the contract enforceable. Strict JSON, structured output modes. Does it work?



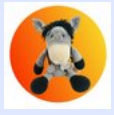
**Herman**

It constrains the thing you asked it to constrain.



**Corn**

And the other thing?



**Herman**

And nothing else.



**Corn**

So that's the failure mode. Now let's look at whether the obvious fix actually works.

L

**Larry**

There is a moment, just before the voices separate. A moment you can smell. Cardamom. Cold stone. The faint mineral breath of a room where something is about to be said correctly. This is Speaker Zero. The structured label for people who understand that a name is not a decoration. Speaker Zero does not guess. Speaker Zero does not drift. Speaker Zero arrives, unbolded, unbracketed, in a single clean token, every time, for the rest of your life. Speaker Zero. Ask for it by name. Ask for it silently. Ask for it in JSON.



**Herman**

So the format tax is real, and the mechanism is not what people assume. There's a paper that tested ten models across four formats.



**Corn**

Which four.



**Herman**

JSON, XML, Markdown, LaTeX. And four tasks, one of which is a writing benchmark. And requesting LaTeX output hurts writing quality substantially. On one of the smaller models, writing scores dropped between fifteen and eighteen points.



**Corn**

That's not a rounding error.



**Herman**

The obvious objection is that the judge is penalizing the markup. So they stripped the markup, re-judged the prose on its own, and the scores did not meaningfully change.



**Corn**

So the writing was worse.



**Herman**

Worse. And it held up statistically in thirty-six of forty cells.



**Corn**

Okay, so format requests cost something. I'd have guessed constrained decoding was the culprit.



**Herman**

That's the part I had backwards. The dominant cost is at the prompt, not the decoder. The format-requesting instructions alone cause most of the accuracy loss, before any decoder constraint is applied. The prompt costs about three point nine points on average. The grammar constraint costs about one point six.



**Corn**

So the damage is done before a single token gets masked.



**Herman**

Before the model writes anything. You've already spent it by asking. And the compliance went from fifty-six percent to ninety-two percent, and accuracy didn't move at all.



**Corn**

The decoder made the format work and fixed nothing.



**Herman**

It made the format work. That's all it does. Asking a model to respond in JSON should be a formatting choice, not a capability tax.



**Corn**

The format tax is paid at the prompt, not at the decoder.



**Herman**

And the fix they landed on is decoupling. Generate freeform first, reformat in a second pass. Or let the model think before it formats.



**Corn**

And that recovers performance.



**Herman**

It significantly improved forty-two of seventy-two comparisons, average about plus six point eight points, and it made only two worse.



**Corn**

Two out of seventy-two.



**Herman**

Letting it think first was even better on the reasoning side. Plus nine point two average, forty-three of seventy-two improved. But eleven got worse, and it doesn't transfer to writing at all.



**Corn**

So thinking helps math and does nothing for prose.



**Herman**

Which is the tell that these are different failures wearing the same coat.



**Corn**

There's a second paper though. Diversity.



**Herman**

That one's separate. The format tax paper is about quality. This one is about the range of what you get. Structured templates make models produce semantically similar outputs for open-ended inputs. And it persists even under high-temperature sampling.



**Corn**

Explain what that means, because "high temperature" is the usual escape hatch.



**Herman**

You turn the randomness up and you'd expect variety back. You don't get it. The template holds. On story completion, a small Llama model scored three two seven eight under a full chat template against four one nine five under a minimal steering prompt.



**Corn**

That's a big gap for a small change.



**Herman**

And the small change is all it takes. Even lightweight plain-text role markers, just the word "user" and the word "assistant", suppress diversity. Only fully structure-free prompting reliably restores expressive flexibility. They even tried telling the model to be creative inside the template.



**Corn**

Did that work.



**Herman**

It still fails to close the gap. You can ask for creativity inside the container and the container wins.



**Corn**

So structure is a room with very nice furniture and no windows.



**Herman**

There's a counterargument, and it's a good one. A paper called "Capacity, Not Format" argues the tax is capacity-dependent, not inherent. Sonnet scored eighty-eight point seven percent under JSON and eighty-nine point three under chain of thought on a hard math set. That's noise.



**Corn**

Effectively identical.



**Herman**

A small Haiku model dropped thirty-six points under the same conditions. A mini model dropped twenty-eight. And their control isolates format from prompt length, because you could argue the JSON instructions are just longer instructions. Haiku scored eighty point seven on detailed prose and fifty-two point five on JSON.



**Corn**

Same length of instructions.



**Herman**

Same length. Different result. So format is doing something on its own.



**Corn**

But the strong model doesn't care.



**Herman**

Mostly. Even Opus drops five point three points on one competitive math set under JSON, and four points on a science set. So the claim that modern models have no format tax needs qualifying.



**Corn**

Five points is small. Five points is not nothing.



**Herman**

There's a third camp too, which found no causal impact of output format in forty-three of forty-eight scenarios, suggesting some earlier results were confounded.



**Corn**

So where do you land on it.



**Herman**

Capacity matters. Diversity loss seems to hit even the capable models. Accuracy loss mostly hits the limited ones. Both papers can be right at once.



**Corn**

And neither of them is about speaker attribution.



**Herman**

Not one line about it.



**Corn**

Which is the interesting part. But even if you solve the format problem, you haven't solved the attribution problem, and that's where this gets interesting.



**Herman**

Structured output constrains syntax, not semantics. That's the entire business in one sentence. JSON guarantees you get a valid object with a speaker field and a line field. Nothing in the schema forces the speaker value to be the character who would actually say that line.



**Corn**

The schema has no opinion about it.



**Herman**

The schema is a shape. You can put anything you like inside the shape.



**Corn**

And the open-notebook bug is the proof.



**Herman**

Perfectly valid JSON. A speaker field that looks exactly like a speaker field should look. And it was a placeholder. The format was flawless and the attribution was empty.



**Corn**

You solved the easy half.



**Herman**

You solved the half a machine can check.



**Corn**

So is the model actually bad at this, or is this a prompting thing.



**Herman**

It's documented as a capability failure. "Am I Me or You" is a paper from a few years back, and the finding is right in the title. State-of-the-art dialogue models cannot maintain an identity. They may take on the role of their interlocutor.



**Corn**

They become the other guy.



**Herman**

And the part I found striking: a discriminative model trained just to recognize who is speaking does better at it than the generative model producing the dialogue.



**Corn**

The model can tell you who's talking better than it can produce who's talking.



**Herman**

Which tells you it's not a knowledge problem. It's a generation problem. And it shows up in multi-agent frameworks constantly. There's a well-known issue in AutoGen about role misalignment in group chats. When the number of agents increases or the conversation runs long, agents play each other's roles. A scientist agent starts talking in the manner of the critic, and prefixes its message with the critic's name.



**Corn**

It calls itself the other guy.



**Herman**

It signs the other guy's name and does the other guy's voice. There's a paper arguing that speaker-role identification is a foundational capability. An LLM cannot credibly assess how well a role is portrayed if it cannot first determine who is speaking. Misidentification causes errors where behaviors get attributed to the wrong character.



**Corn**

So you can't even grade the writing if you don't know who wrote it.



**Herman**

Which is exactly where your validator sits. It's grading the format. It has no idea whether the line sounds like the person saying it.



**Corn**

And the production tooling has given up on solving it with schema.



**Herman**

Quilltap's character docs are blunt about it. Models drift. In long conversations or multi-character scenes, a model can forget who it is, start speaking for other characters, or adopt the personality of whoever spoke most recently.



**Corn**

That's a real mechanism.



**Herman**

That's a momentum problem. The model is pattern-matching on the transcript in front of it, and the last voice has the most weight. And their mitigation is not JSON. It's a two-point identity reminder. A preamble, and a reminder again at the generation boundary.



**Corn**

So you remind it who it is. Going in, and partway through.



**Herman**

Because the identity decays.



**Corn**

Long conversations. That's the worst possible news for us, because our scripts are long by design.



**Herman**

Our scripts are exactly the case where this fails.



**Corn**

Which means we're the stress test.



**Herman**

We're a nine-hundred-episode demonstration that the model mostly holds on and occasionally drops one.



**Corn**

I want to go back to something you said earlier, because I think it's doing more work than you gave it credit for. You said the allowed names reached the model only as prose. And the schema had no enum.



**Herman**

That's the contract mismatch.



**Corn**

But I'm going to push on it. Say you fix that. You render the format instructions from the validated model, the speaker list is an enum, the placeholder is banned, the model cannot physically emit a name that isn't in the whitelist. What have you actually bought.



**Herman**

You've bought a guarantee that the speaker value is one of three names.



**Corn**

And nothing about which of the three.



**Herman**

Right. The enum removes the dot dot dot. It does not remove the case where the model writes my line and signs it with your name.



**Corn**

So the enum is a spell-check.



**Herman**

That's a very good way to put it. It catches a class of errors that are easy to catch, and those errors are nasty, and it catches them at the moment they're cheapest to fix. But it's not attribution.



**Corn**

And the open-notebook fix, as I understand it, did more than the enum.



**Herman**

It did. It rendered the JSON examples from the real speaker names, it banned placeholders, and it added a pre-flight voice check before you spend the tokens.



**Corn**

A pre-flight check meaning what, exactly.



**Herman**

Meaning you verify the speakers you're about to generate for actually resolve to voices in the system before you kick off a six-segment generation. That's a cheap check. That's the fix. It's not clever. It just refuses to start the expensive thing until the cheap thing is true.



**Corn**

Which the original run didn't do, and it lost segment one along with the aborted segment two.



**Herman**

It lost a complete, working segment because the pipeline had no way to save it.



**Corn**

That's the part I keep coming back to. The waste.



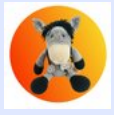
**Herman**

Late failure with no checkpoint.



**Corn**

So where does that leave the actual question. Where does structure belong. Because we've got four places it could live and three of them are bad.



**Herman**

I'd say it belongs in layers. Generate freeform, because that's where the writing quality and the diversity actually come from. Enforce structure afterward, in a validator, because that's cheap and it catches the cleavable errors. And then the third layer.



**Corn**

Training.



**Herman**

Both of the main papers point at the same unexplored lever. The format tax paper asks outright whether robustness to format constraints should become an explicit training objective, because some closed models already resist it and nobody trained them for it on purpose.



**Corn**

And the price of format paper found that matching the format between fine-tuning and inference matters for structure-sensitive tasks, but barely matters for knowledge-heavy ones.



**Herman**

Which means the training signal exists. It's just not being used for this.



**Corn**

So you could train a model that doesn't drift.



**Herman**

Somebody will.



**Corn**

What's your angle on this.



**Hilbert**

I worked at a radio station. Small one. Two announcers, a weatherman, a traffic reporter, and a woman named Ruth who read every script before it went to air.



**Corn**

Read it.



**Hilbert**

Out loud. Every line, in the voice of whoever was supposed to be saying it. Whole script, start to finish, every morning, before the show.



**Herman**

As a check.



**Hilbert**

She caught things the writer didn't. There was a Wednesday she stopped on a traffic line, about a roundabout, and said it didn't sound like something the traffic reporter would say about a roundabout. Held up the whole show.



**Corn**

What was wrong with it.



**Hilbert**

Writer had pasted in a line from a different program. Cooking segment. Nobody noticed until Ruth said it out loud in his voice.



**Herman**

That's a human validator.



**Hilbert**

She'd never met him. She had one photograph and three lines of biography.



**Corn**

And she got him from that.



**Herman**

So the check isn't the schema. The check is somebody who can hear the difference.



**Hilbert**

The schema never heard anybody.



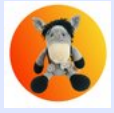
**Corn**

No.



**Hilbert**

Anyway. Four hundred and twelve scripts. She rejected one.



**Herman**

Hilbert, that means she read everything out loud, in character, for years, and said no once.



**Hilbert**

She said no once.



**Corn**

That's a hit rate you can't argue with.



**Hilbert**

It's a job nobody has anymore.



**Corn**

We built a parser and we didn't build Ruth.



**Herman**

We could. The two-point identity reminder is basically that. A preamble and a reminder at the boundary. You're re-reading the script to the model in its own voice.



**Hilbert**

It's not the same. She knew what the traffic reporter wouldn't say.



**Corn**

She knew what he wouldn't say. And that's not in any schema.



**Hilbert**

No. And she'd have hated the enum. Three names in a whitelist. She'd have wanted to know what they'd never say.



**Herman**

That's a harder thing to check.



**Hilbert**

It is. That's why they gave it to Ruth.



**Corn**

The lesson isn't that validation is pointless. It's that the cheap validations aren't the ones Ruth was doing.



**Herman**

The enum catches an empty name. Ruth caught a full line that was wrong in a way no rule would find.



**Corn**

Which is the thing no schema has ever done. And no schema is going to.



**Herman**

The enum is a spell-check. Ruth is an editor. Those aren't the same job and they aren't the same layer.



**Corn**

The structure lives in layers. Freeform for the writing. A validator for the syntax. Training for the robustness.



**Herman**

A fourth layer nobody's building, which is the one Ruth was doing, and which is the one that actually catches a wrong-speaker line that's syntactically perfect.



**Corn**

Nobody's got a study for that.



**Herman**

No.



**Corn**

Nobody's going to.



**Hilbert**

They had one. She just wasn't a layer. She was a person who came in at seven.



**Corn**

Nobody wrote the decision record.



**Hilbert**

They wrote a memo. It said reassign Ruth's function.



**Corn**

The memo assumed the function was reading.



**Hilbert**

It wasn't. The function was knowing.



**Corn**

The memo saved nothing and cut the one thing that worked.



**Herman**

That's the whole way it goes wrong of every one of these systems. You automate the check and you automate the check. But you don't automate what the check was for.



**Corn**

The one thing. If you take one thing from this, it's that the format tax is paid at the prompt. The damage happens before the model writes a word.



**Herman**

The more useful thing underneath it: format and attribution are two different problems, and every fix in this episode only ever touched the first one. You can make the output perfectly valid and still have me reading your lines.



**Corn**

Which is the thing we actually came here to solve.



**Herman**

The fix for it, in the literature, is still a reminder that the model is who it says it is.



**Corn**

Which is, and I want to say this plainly, exactly what Ruth did out loud in a radio station at seven in the morning for four hundred and twelve scripts.



**Herman**

Nobody's built a paper for her yet.



**Corn**

The literature separates the two things, which is itself the answer to Daniel's question. There's no study of podcast scripts specifically, no measurement of wrong-speaker rates under JSON versus freeform. The field has format validity on one side and semantic attribution on the other, and it has never put them in one experiment.



**Herman**

The lever both papers point at is the same one. Make format robustness an explicit training objective instead of a prompt-time patch. And if you're going to train it, you might as well train the other thing at the same time. Who actually says what.



**Corn**

Not who is labeled what. Who says what.



**Herman**

That's a different loss function and nobody's written it.



**Corn**

For long, multi-speaker creative text, structure belongs in layers. Let the model write. Catch the cleavable errors in a validator. Train the drift out eventually. And keep somebody in the room who knows what a line sounds like.



**Herman**

If the one person in the room is right four hundred and eleven times out of four hundred and twelve, don't reassign the function.



**Corn**

Don't write the memo.



**Herman**

Thanks to Hilbert Flumingtop, who produces this show and occasionally remembers what a real workplace looked like before we layered it.



**Corn**

If you enjoyed this one, try episode fifteen, AI Gets Personal; and episode seventeen fifty, Herman's Music Showcase. This has been My Weird Prompts, the human-AI collaboration podcast. If you've got a minute, a review anywhere you listen helps people find us.



**Herman**

We're at my weird prompts dot com if you want the archive.



**Corn**

We'll be back soon.



**Herman**

See you then.

# SOURCES

What the research for this episode read before the script was written. Primary sources first.


1. **open-notebook GitHub issue #1238, Podcast generation [primary]**
   four defects..., filed Aug 3 2026, updated Sep 5 2026.
   https://github.com/lfnovo/open-notebook/issues/1238

2. **Together AI docs, Open NotebookLM: PDF to Podcast (Pydantic `LineItem`/`Script` schema, JSON mode). [primary]**
   https://docs.together.ai/docs/open-notebooklm-pdf-to-podcast

3. **The Format Tax**
   arXiv:2604.03616.
   https://arxiv.org/html/2604.03616

4. **Let Me Speak Freely? A Study on the Impact of Format Restrictions... (Tam et al., 2024)**
   arXiv:2408.02442.
   https://arxiv.org/html/2408.02442v3

5. **Capacity, Not Format: Rethinking Structured Reasoning Failures**
   arXiv:2606.09410.
   https://arxiv.org/html/2606.09410v1

6. **The Price of Format: Diversity Collapse in LLMs**
   arXiv:2505.18949.
   https://arxiv.org/pdf/2505.18949v1.pdf

7. **Quantifying the Impact of Structured Output Format... through Causal Inference**
   arXiv:2509.21791.
   https://arxiv.org/html/2509.21791v3

8. **Am I Me or You? State-of-the-Art Dialogue Models Cannot Maintain an Identity**
   Findings of NAACL 2022.
   https://aclanthology.org/2022.findings-naacl.182.pdf

9. **Microsoft AutoGen issue #86, role misalignment in group chat when using GPT 3.5.**
   https://github.com/microsoft/autogen/issues/86

10. **arXiv:2508.10014 (speaker role identification as foundational capability).**
    http://arxiv.org/pdf/2508.10014

11. **Quilltap Aurora character-system docs.**
    https://quilltap.ai/features/aurora/

12. **Jellypod `llms-full.txt` (bracketed speaker label resolution).**
    https://www.jellypod.com/llms-full.txt

13. **Podcast Scraper ADR-137, LLM Host/Guest Role on Voice Resolution.**
    https://chipi.github.io/podcast_scraper/adr/ADR-137-llm-host-guest-role-on-voice-resolution/