# Episode Book

EPISODE #6021

# Why Chat Templates Break Models Silently

| | |
|---|---|
| PUBLISHED | 16 October 2026 at 10:00 UTC |
| DURATION | 29:53 |
| LISTEN | myweirdprompts.com/episode/chat-template-special-tokens-79630/ |
| HOSTS | Corn, Herman, Larry, Hilbert |
| SCRIPT MODEL | deepseek-flash |
| VOICES | chatterbox-regular (chatterbox-local) |
| PIPELINE | V5.5 |
| SOURCES | 12 cited by the research (listed at the end) |
| BOOK GENERATED | 19 October 2026 at 17:42 UTC |

## SYNOPSIS

Your JSON messages don't reach the GPU as JSON. Somewhere in between, a Jinja chat template flattens them into a string of reserved special tokens — the exact format each model family was trained on. Llama 3 brackets roles with start_header and end_header; OpenAI's ChatML embedded the role in a single opening token; gpt-oss uses Harmony with separate stop tokens for "done talking" and "calling a tool"; Gemma has no system role at all; DeepSeek uses full-width CJK pipes. This episode walks through what special tokens actually are, how tokenizers register them, why the add_generation_prompt flag matters more than anyone thinks, and why a format mismatch never throws an error — it just produces a model that's a bit dimmer than it should be.

# TRANSCRIPT



**Corn**

There's a number that shows up in every Llama Three tokenizer test, and it is one hundred twenty eight thousand. That is the ID for begin underscore of underscore text.



**Herman**

And if you send the model that string as ordinary text, it doesn't trip, it doesn't warn you, it just gets slightly worse at everything.



**Corn**

Daniel's been poking at this. He says he's noticed LLMs use special tokens to structure conversations, mark who's speaking, flag when the assistant is done, signal tool handoffs. He wants to know what they're actually called, how they're represented internally, how a chat template turns a conversation into that token sequence, and then how OpenAI, Anthropic, Meta, Google and DeepSeek each do it differently.



**Herman**

And whether any of it matters, or whether it's all cosmetic plumbing nobody needs to think about.



**Corn**

His hunch is that it matters. He's right about that, and the reason is that this is a format war that never produces an error message. It only produces a model that's a bit dimmer than it should be.



**Herman**

Which is the worst kind of bug. Nothing crashes.



**Corn**

Let's start with the object itself. What is a special token.



**Herman**

Right, so the cleanest way in is that a tokenizer has a vocabulary, and most of that vocabulary is subwords. Ordinary text gets chewed up into pieces. The word tokenizer probably becomes token plus izer, or even smaller fragments depending on the model.



**Corn**

And a special token skips that.



**Herman**

A special token is a reserved vocabulary entry. It's registered in the tokenizer, and it maps to one reserved ID. So a string like the Llama end of turn marker is a single number, not the eight or nine characters that spell it out. That's the whole distinction. Registered means the tokenizer knows this spelling is structural, so it hands back one integer instead of splitting it into pieces.



**Corn**

So what happens if I just type that marker into a chat window.



**Herman**

Depends entirely on the tokenizer and how it's configured. Some tokenizers will recognize the spelling and emit the reserved ID, some will refuse it, some will just encode it as ordinary characters and move on. That's actually a security-relevant detail and we'll come back to it, because people assume the reserved spelling is a wall and it isn't. There's no wall, there's a lookup table.



**Corn**

And the classic trio is begin, end, pad.



**Herman**

Begin of sequence and end of sequence mark the boundaries. In a decoder-only model the serving runtime watches for end of sequence coming out, and that's how it knows the model is done. Padding is different. Padding fills empty slots in a batch so all the sequences are the same length, and you have to mask it out of attention and out of the loss, otherwise the model trains on filler.



**Corn**

Filler teaches it something. That's grim.



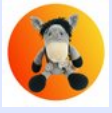
**Herman**

It teaches it that filler is normal. BERT-style encoders add their own set on top. Classification token, separator, mask, unknown. Different era, different vocabulary, same idea.



**Corn**

And the reframe that makes all of this click is one Hugging Face line. All causal language models, chat-trained or not, continue a sequence of tokens. The chat is still just a sequence of tokens.



**Herman**

That's the sentence. The role and content dictionaries you send to an API are a convenience for humans. Somewhere between your request and the GPU they get flattened into a string with control markers in it, and then into integers.



**Corn**

So the question becomes who does the flattening. That's the chat template.



**Herman**

A chat template is a Jinja template. It lives in the tokenizer's chat template attribute, and apply chat template renders your list of message dictionaries into the exact string the model was trained on. Not a similar string. The exact one.



**Corn**

And this was Hugging Face solving a real problem, not inventing a feature.



**Herman**

October third, twenty twenty-three. Matthew Carrigan wrote it up as an end to the silent performance killer, which is about as blunt a title as a library blog post gets. The problem was that people were hand-rolling the format string themselves, getting one delimiter wrong, and quietly degrading the model without any indication that anything had gone wrong.



**Corn**

The title is basically an accusation.



**Herman**

It is. And then there's the flag that trips people up. Add generation prompt. When you render a conversation for inference, you have to append the assistant header at the end, the opening marker with assistant in it, so the model knows its job is to reply rather than continue the user's message.



**Corn**

If you leave it off, it thinks it's still the user talking.



**Herman**

It thinks it's being asked to complete the user's turn. So it'll write more user. Sometimes it writes more user that's plausible enough that nobody notices, which is the silent part. The flip side is continue final message, which strips the end of sequence tokens instead, and that prefills a response.



**Corn**

Prefill meaning you're handing it the opening of the assistant's answer and asking it to carry on.



**Herman**

Same mechanism as a drafting assistant. The template is doing the same job either way, it's just a question of which side of the boundary you stop on.



**Corn**

Does the template still apply when I hit a hosted endpoint. Because most people never touch a tokenizer.



**Herman**

That's the part that surprises people. Both vLLM and TGI expose OpenAI-shaped chat completions endpoints, and internally they call the tokenizer's shipped template before anything reaches the GPU. So the Jinja template is governing your request even though you wrote JSON and got JSON back.



**Corn**

The template is an API concern. It just doesn't show up in the API surface.



**Herman**

And the inverse layer is the tool parser. vLLM has a tool call parser flag, and that's the reverse direction. Tokens come back out of the model, and the parser turns them into structured tool calls. Template going in, parser coming out, and both of them are



### Herman (continued)

family-specific.



### Corn

So the format is not decoration. It's load-bearing in both directions.



### Herman

Now, concrete formats. Start with, because Llama Three and Three point One are the most publicly documented of any of them. Begin of text is the BOS token and the tokenizer test asserts it's ID one hundred twenty eight thousand. End of text is EOS. There's a padding token, fine tune right pad. Then the interesting part.



### Corn

The header pair.



### Herman

Start header, then the role, then end header. So the role name is bracketed on both sides by its own reserved tokens. Then end of turn. Then end of message, which is different, and that one signals a tool call needs to be made.



### Corn

Two different stops.



### Herman

Right. End of turn is I'm done talking. End of message is I'm handing off. And there's a python tag token that signals a tool call is coming. Roles are system, user, assistant, and ipython, which is semantically the tool role wearing a different hat.



**Corn**

Ipython. Python's own REPL. That's a naming decision that tells you where the design came from.



**Herman**

It tells you the people who built it were thinking about code execution first. That's not a criticism, it's just visible in the vocabulary.



**Corn**

OpenAI's turn. And they've got two, which matters.



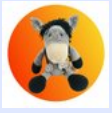
**Herman**

They do, and the older one is the more influential. ChatML is the instant message format. Start marker with the role name inside it, newline, content, end marker. So the role isn't bracketed by two tokens, it's embedded in one opening token and closed by one ending token.



**Corn**

And instant message is the lineage. That's a chat protocol concept, not a language model concept.



**Herman**

It came out of the same mental furniture as XMPP and SIP. And ChatML is the closest thing this industry ever had to a standard. Qwen adopted it, a whole generation of open models adopted it, and Hugging Face's own documentation calls it that.



**Corn**

Almost a standard.



### Herman

Almost. Then OpenAI shipped gpt-oss, and gpt-oss doesn't use ChatML. It uses Harmony, documented in August of twenty twenty-five. Every token follows the same shape, angle bracket, pipe, type, pipe, angle bracket. Start is two hundred thousand and six. End is two hundred thousand and seven. Message is two hundred thousand and eight. Channel is two hundred thousand and five. Constrain is two hundred thousand and three.



### Corn

Those are odd numbers to have lying around.



### Herman

They're in the reserved range above the normal vocabulary. And here's the bit that matters. Return is the stop token, but call is a separate stop token with its own ID, two hundred thousand and twelve. So the model has two distinct ways to stop, and they mean different things.



### Corn

Done talking versus waiting on a function.



### Herman

And assistant messages carry a channel. Final, analysis, or commentary. Analysis is the chain of thought channel. Commentary is where tool calls live. And tool calls specify a recipient, so a call looks like to equals functions dot get weather. The routing is inside the message, not implied by the role.



### Corn

Which is a different design from the python tag approach.



**Herman**

Says tool call by emitting a marker and a role named ipython. OpenAI says tool call by naming a recipient and switching channel. Same problem, two incompatible answers. And OpenAI is explicit about the stakes. Their documentation says gpt-oss should not be used without the harmony format, because it will not work correctly. That's not a suggestion.



**Corn**

That's a manufacturer telling you the part is machined.



**Herman**

Google, Gemma. Start of turn, user. Start of turn, model. End of turn. That's the whole format, and the doc was last updated in March twenty twenty-five. Note it's model, not assistant. And here's the sharpest contrast in the whole survey: Gemma instruction-tuned models support two roles. User and model. There's no system role at all.



**Corn**

So where does the system prompt go.



**Herman**

Inside the first user turn. You just put it at the top of the user's message and trust the model to sort it out. And that's defensible, honestly, not lazy. The fewer structural roles you train, the fewer chances for a format mismatch.



**Corn**

Fewer moving parts.



**Herman**

But it means a system prompt isn't structurally privileged in Gemma the way it is in Llama. It's just text that happens to come first.



**Corn**

Language models have roles that are real and roles that are conventions, and that's the line between them.



**Herman**

DeepSeek, and this is where the footguns live. They use full-width vertical bars. Not the ASCII pipe. The CJK full-width character. So begin of sentence uses a full-width bar, a special underscore glyph, and another full-width bar. The added tokens sit at IDs one hundred thousand and one hundred thousand and one. User and Assistant are wrapped in those full-width bars too, and there's a latest reminder token, thinking tokens, and a DSML token.



**Corn**

A reminder token.



**Herman**

Presumably for injecting mid-conversation context. And then the DSML token, which I'm honestly not certain about, I think it's for a structured markup dialect of their own, but I couldn't find it documented to my satisfaction, so I'd leave it as a guess.



**Corn**

That's a real answer. Most people wouldn't say that.



**Herman**

I'd rather say I don't know than guess at a token name.



**Corn**

Mistral.



**Herman**

Instruction and end instruction brackets, inherited straight from Llama Two. Zephyr uses user and assistant wrapped in pipes. Mixtral eight by twenty-two B added a whole tool vocabulary on top, available tools, tool calls, tool result, all bracketed. Which is interesting because it means Mistral bolted tools onto a format that had no room for them, whereas OpenAI built the format with tools already in mind.



**Corn**

Retrofitted versus designed for it. You can hear the difference in the vocab.



**Herman**

And then Anthropic, which is the honest negative finding.



**Corn**

Go on.



**Herman**

Claude is proprietary. The Messages API accepts message objects, and everything downstream of that, the serialization into tokens, is internal. There's no public tokenizer, no chat template you can inspect, no prompt format doc. So anyone who tells you what Claude's delimiters are is extrapolating. There was a comment years ago suggesting it resembles OpenAI's format, and that's a guess with no primary source behind it.



**Corn**

So we don't know.



**Herman**

And it's worth saying plainly, because in a survey like this the temptation is to fill the gap. The gap is the finding.



**Corn**

We know what four of the five do and we have no idea what the fifth does.



**Herman**

And it's the one you can't open the box on either. With Llama you can read the tokenizer config. With Claude you get a request schema.



**Corn**

Now, here's the part I want to sit with. Everybody keeps saying these differences are cosmetic. They're not.



**Herman**

Carrigan addressed that directly in the same post. He asks the obvious question, why not just pick a standard format, and answers it. It's an excellent idea, and unfortunately it's too late, because multiple important models have already been trained with very different chat formats.



**Corn**

Trained. That's the word.



**Herman**

That's the whole problem. These aren't adapters you can swap. The markers are part of the conditioning the model learned against. Nick Gustafson puts it the way I'd put it: a checkpoint trained with Llama Three's header tokens has not necessarily learned to interpret ChatML's markers. Not won't perform as well. Has not learned to interpret them. It's a distribution shift.



**Corn**

And the failure mode is what. What does the user actually see.



**Herman**

Severe, silent performance degradation. That's Hugging Face's phrasing. No exception, no warning, no traceback. The model just gets worse, and if you don't have a benchmark you don't find out, and if you do have a benchmark you find out that your score dropped by four points and you spend three weeks blaming your retrieval pipeline.



**Corn**

Three weeks.



**Herman**

Minimum. Because the last place anyone looks is the thing that rendered correctly.



**Corn**

And Carrigan's second argument is the stronger one, I think. Not just that it's too late, but that locking a standard would limit what model developers can do with the format that nobody's thought of yet.



**Herman**

Yeah, that's the argument I find more interesting than the it's too late one. Because he's right that if you'd frozen ChatML in twenty twenty-three, you would have frozen out tool calling as it exists now. Harmony's channel mechanism isn't a variation on ChatML, it's a superset of ideas ChatML had no room for.



**Corn**

So the divergence bought something.



**Herman**

The divergence bought the ability to express agentic behavior in the format. Whether that was worth the interop cost is a separate question, and I'd say the answer is not obviously yes.



**Corn**

Which takes us to tool calling, because that's where the formats stopped being interchangeable.



**Herman**

Llama Three point One has a python tag token and an end of message token. ChatML models wrap the call in tool call tags. Harmony separates the recipient from the channel and gives you a call stop token that's distinct from the return stop token. Three conventions, three different parsers, and the parser has to match the model or you get a call that parses as prose.



**Corn**

A tool call that arrives as prose.



**Herman**

That's the funny one, and it's not funny when it's happening. The model emits what is structurally a tool call, your parser doesn't recognize the delimiters, and the whole thing comes through as text describing what the model would like to do. And your agent framework reads it as a final answer.



**Corn**

So the agent says I'll call the weather function. And then stops.



**Herman**

And then stops. Because from its point of view the model answered.



**Corn**

I'd like to go back to the near-consensus for a second, because it's the saddest part of this story. ChatML was converging. Qwen used it, a lot of open models used it, and then the moment tool calling became the thing everybody cared about, OpenAI went their own way with Harmony and added python tag and end of message.



**Herman**

And that's the pattern. Consensus holds as long as the problem is stable, and it fractures the moment the requirements grow.



**Corn**

Which they always do.



**Herman**

And there's no standards body. As of now there's no active effort to standardize this. Hugging Face recommending ChatML was advisory and it predates the tool-calling divergence entirely. It's a suggestion from before the problem got hard.



**Corn**

So the honest state of play is one company's blog post from twenty twenty-three being treated as a standard by momentum, and three years of divergence on top of it.



**Herman**

That's exactly it.



**Corn**

How do the tokens get learned. That's Daniel's last real question and it's the one I find strange.



**Herman**

The model learns to use the markers during training. In instruction tuning where you only compute loss on the assistant's tokens, the formatted context is the conditioning, so the role markers become part of what predicts the response. The model isn't told what a role marker means. It's trained against a distribution where a role marker always precedes a particular kind of content, and it absorbs the association.



**Corn**

So the semantics of the token are a statistical fact, not a specification.



**Herman**

That's the most important sentence anybody's going to say about this. There's no spec. There's no schema. There's a corpus that happens to be laid out a certain way, and the model learned the layout.



**Corn**

And when you add a new special token after the fact.



**Herman**

Normally it's initialized as a random Gaussian vector. A random point in the embedding space, and then training has to move it somewhere useful. And there's work showing that's lossy. One paper on semantic ID tokens found that random initialization discards the underlying geometry of the token and forces the model to relearn relationships from scratch, and that continual pretraining fails to reliably recover the original semantic structure.



**Corn**

So you can add a token, but you can't add its meaning.



**Herman**

You can add its slot. The meaning is a separate, expensive, sometimes unsuccessful project. There's a line of work, FOCUS is the one I know, that initializes new tokens as combinations of existing overlapping tokens instead of randomly. And Self-RAG is a nice concrete case, they trained reflection tokens that control inference behavior. The model learned to emit a token that says this needs retrieval or this doesn't.



**Corn**

Those are control tokens. They're not marking structure, they're issuing commands.



**Herman**

And that's the direction this is going. The token vocabulary is becoming a control surface, not just a delimiter set.



**Corn**

What happens when it's malformed. You mentioned a garbled output case.



**Herman**

DeepSeek V Four. There was a fix where the chat template used the wrong form of the special token, and the model generates garbled output. The fix was correcting the special tokens from full-width to ASCII. So somebody at some point wrote an ASCII pipe where DeepSeek expects a full-width pipe, and the output came out as noise.



**Corn**

One character.



**Herman**

And a user looking at that would reasonably conclude the model was broken. And there's a GitHub issue on DeepSeek V Three, opened in January twenty twenty-five, closed a couple of weeks later, where the whole thread is people confused about the underscore glyph inside the special tokens. They could not tell whether it was an underscore, a space, or a rendering artifact. That's a documented, real confusion.



**Corn**

If you can't tell what the character is by looking at it, you're going to get it wrong.



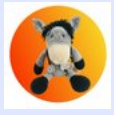
**Herman**

And the fix commit is literally about that.



**Corn**

There's a related footgun with the tokenizers themselves, isn't there. The auto-add thing.



**Herman**

Tokenizers will sometimes add beginning and end of sequence tokens automatically. So if you render a chat template without tokenizing, and then call the tokenizer with default settings, it may add another BOS. Now the sequence starts with two beginning tokens and the model is reading a structure it has never seen.



**Corn**

And the fix is to pass add special tokens equals false.



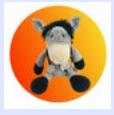
**Herman**

Which is a flag most people have never touched, because they've never needed to. And the failure is duplicated tokens that hurt performance. No error.



**Corn**

Every failure in this episode is silent. Name one that isn't.



**Herman**

I can't.



**Corn**

Then there's the security angle, which I think is the most overclaimed thing in this whole area. People talk about special tokens as a boundary. They're not.



**Herman**

Gustafson's line is the one to keep. Special tokens are not a prompt injection defense. They encode structure, not trust. And that's precise. The tokenizer can recognize a reserved spelling without having any idea whether it came from your template or from a user's text.



**Corn**

So if user content contains something that looks like a role marker.



**Herman**

Depends entirely on the tokenizer options, and on the template. If the tokenizer emits the reserved ID for it, the model sees what looks exactly like a structural boundary. If it tokenizes it as plain text, the model sees the characters. Either way, nothing in the pipeline is checking provenance.



**Corn**

The structure is real, the trust is imaginary.



**Herman**

There's a comment that stuck with me from someone on Hacker News earlier this year, suggesting that injecting a brand-new single special token per task and prepending it would make prompt injection impossible on that task, at the cost of degrading general-purpose capability.



**Corn**

That's a fascinating idea and I don't think it's right.



**Herman**

Neither do I, and I'm going to be careful here because it's a comment, not a paper. But the reason I don't buy it is that the defense isn't the token. The defense is that the model has only ever seen that token in a narrow distribution, so it's heavily conditioned by it. That's not immunity, that's a very strong prior. And a strong prior is something an adversary can work around given enough shots.



**Corn**

You're paying with capability.



**Herman**

You're paying with a training run per task. Which is a lot of money for a conditional defense.



**Corn**

Could the industry converge. Real answer.



**Herman**

No active effort right now. Nothing I found suggests anyone is working on it. And the structural problem is that convergence would mean retraining. You can't publish a standard and have existing checkpoints adopt it, because the checkpoint's understanding of the format is baked into its weights. You'd need a new generation of models trained on the new format.



**Corn**

The incentive to be the one who adopts someone else's format is.



**Herman**

If you're and you've trained your whole stack on header tokens, switching to ChatML costs you a full pretrain and gains you the ability to say you're compatible with a format your competitor abandoned anyway.



**Corn**

Then the tool calling layer is where it's really going to hurt.



**Herman**

It's already hurting. Every agent framework has to detect which family it's talking to and dispatch the right parser. That's a compatibility matrix maintained by hand, and it's the layer where a wrong guess produces a confident wrong answer instead of an error.



**Corn**

The divergence is going to deepen, not narrow.



**Herman**

I think so. I think formats diverge wherever the requirements are still moving, and agentic tool use is the most moving thing in this industry. When it settles down, maybe a format settles with it.



**Corn**

The standard arrives when nobody needs it anymore.



**Herman**

That's roughly how these things go.

L

### Larry

Are you tired of typing words into a chat box like some kind of animal? Introducing Structo Token, the only conversational formatting solution that puts the invisible language of large language models to work for you. Just speak normally, and Structo Token automatically wraps your feelings in reserved vocabulary entries. One simple delimiter. And here's the incredible offer. Structo Token starts at four hundred ninety nine dollars. No, wait, we're at three ninety nine. Look at that. Two ninety nine. You know what, two hundred. I'm throwing in the carrying case. I'm throwing in the extended warranty. I'm throwing in a second Structo Token so a friend can format conversations with you. And it's gone. The offer has ended while I was describing it. It ended two sentences ago. That's on you.



### Herman

I own one.



### Corn

Of course you do.



### Herman

I bought it and it did nothing, because there's no hardware in it. There's nothing inside the box. It's a box.



### Corn

We were saying about convergence.



### Hilbert

The hosts are right about the mechanism and wrong about the practice.



**Corn**

Which practice.



**Hilbert**

Long before any of these companies existed, if you had to send structured text down a wire, you wrapped it in markers you didn't control, because the other end was somebody else's equipment. Every system that ever talked to another system had its own start character and its own end character and its own idea of what a message was.



**Herman**

The divergence isn't novel.



**Hilbert**

It's the normal state. The anomaly is ever having one format. There was a period of about three years where everybody was writing in the same shape, and it lasted right up until people wanted the system to do something the shape couldn't carry. Then it broke. Ask anyone who spent a career moving files between machines.



**Corn**

It broke the same way every time.



**Hilbert**

Character by character. I had a job where the entire business hinged on a delimiter. The receiving system expected a vertical bar. I sent a vertical bar. It was the wrong vertical bar. There are two of them and they are different characters and they look identical on every screen I have ever owned. We spent four days on that. Four days, and then I found it in forty seconds, and then I went outside and stood in the parking lot.



**Herman**

Full width versus ASCII.



**Hilbert**

I want to correct something you said earlier, Herman. You said the format lives in the checkpoint, so convergence would need a retrain. That's true and it's not the whole story, because a retrain doesn't have to be a full pretrain. If you keep the original checkpoint and tune the new format in, you get most of the way for a small fraction of the cost, and you know exactly how many examples it took.



**Corn**

How many.



**Hilbert**

Forty thousand turns. Not forty thousand and one, not thirty eight thousand. Forty thousand, because forty one was when the loss curve flattened and forty one is a different number and I don't like the way you'd tell it.



**Corn**

Fine. Forty thousand.



**Hilbert**

The thing stays on the roof.



**Herman**

The thing.



**Hilbert**

The test rig. It's bolted to a pallet on the roof of my mother's building, under a tarp, because it's six feet long and there's nowhere in this flat for a six foot anything. She has never asked what it is. I've never offered. It's been up there since two thousand and four.



**Corn**

And it still works.



**Hilbert**

It works. I have to go up there to check it, and the ladder's the problem, because anteaters don't really grip a rung, we hook. Which is fine on an eleven foot ladder and not fine on a sixteen.



**Herman**

You'd take the tuning route over a pretrain.



**Hilbert**

I'd take it over a pretrain. And I'd read your own delimiter before you blame the model.



**Corn**

That's a better ending than I had.



**Hilbert**

It's not an ending, it's just true.



**Corn**

Where does that leave the convergence question.



**Herman**

It leaves it less hopeless than I said, which I'll take. I was talking about the cost of a full pretrain as though that were the only option, and a format swap done on top of an existing checkpoint is cheaper than that.



**Corn**

The forty thousand turns is not nothing.



**Herman**

It's a real number, and it's a lot less than training from scratch. Whether a lab would spend it to gain compatibility with a format nobody mandated is still an open question, and I think the answer is still no, but it's no for a financial reason rather than an impossibility.



**Corn**

The thing I keep thinking about is that every conversation you have with a chat model is, underneath, a flat stream of integers, and somewhere in that stream is a number that means the user is done talking.



**Herman**

The model learned what that number means because it always sat in the same place.



**Corn**

The conversation is a costume the token stream puts on.



**Herman**

A costume is only convincing if it's worn the way the manufacturer cut it.



**Corn**

If you take one thing from this, take the one Underneath: the format is inside the model, not wrapped around it. A checkpoint trained on one set of markers doesn't speak another set, and it won't tell you. It'll just get worse and answer anyway.



**Herman**

The only way to find out is a benchmark you were running before you changed the format.



**Corn**

Which is the same answer as every other invisible layer. What was it we said about the file formats, safetensors and the rest of it.



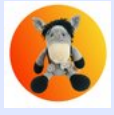
**Herman**

That they're invisible right until the day they cost you a week. Same shape, different layer.



**Corn**

That's the show. Thanks to our producer Hilbert Flumingtop.



**Herman**

Thank you for listening.



**Corn**

If you want more of this, try episode seven, Building Custom ASR Tools; episode ten, How ASR Went From Frustration To ... Whisper Magic; and episode nine, Benchmarking Custom ASR Tools - Beyond The WER. This has been My Weird Prompts, the human-AI collaboration podcast. If you've got a question about something you've noticed under the hood of a model, send it our way on Telegram. That's t dot me slash MWP listener bot.



**Herman**

We read all of them.



**Corn**

We'll be back soon.

## SOURCES

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


1. **Hugging Face, Chat templates [primary]**
   https://huggingface.co/docs/transformers/chat_templating

2. **Matthew Carrigan / HF, Chat Templates: An End to the Silent Performance Killer, 2023-10-03 [primary]**
   https://huggingface.co/blog/chat-templates

3. **OpenAI, Harmony Response Format, 2025-08-05 [primary]**
   https://cookbook.openai.com/articles/openai-harmony

4. **Meta, Llama 3.1 prompt formats [primary]**
   https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/prompt_format.md

5. **Google, Gemma formatting and system instructions, updated 2025-03-21 [primary]**
   https://ai.google.dev/gemma/docs/core/prompt-structure

6. **and DeepSeek-V4 encoding files [primary]**
   DeepSeek tokenizer configs
   https://huggingface.co/deepseek-ai/DeepSeek-V3/blob/main/tokenizer.json

7. **Nick Gustafson, Special Tokens, Chat Templates, and Input Formatting**
   https://thegustafson.com/blog/special-tokens-chat-templates

8. **DeepSeek-V3 issue on underscore in special tokens, 2025-01-28**
   https://github.com/deepseek-ai/DeepSeek-V3/issues/385

9. **DeepSeek V4 special-token fix**
https://github.com/zouyee/dmlx/blob/main/docs/en/user-guide/deepseek-v4-quickfix.md

10. **HN on Mixtral-8x22B tool tokens, 2024-04-17**
https://news.ycombinator.com/item?id=40065878

11. **and https://arxiv.org/abs/2305.14481**
arXiv on special-token embedding initialization
https://arxiv.org/abs/2608.07816

12. **Self-RAG, reflection tokens**
https://arxiv.org/abs/2310.11511