Episode Book

EPISODE #5623

When Your AI Cites Sources It Didn't Actually Use

PUBLISHED	28 September 2026 at 12:36 UTC
DURATION	28:53
LISTEN	myweirdprompts.com/episode/ai-citation-faithfulness-tracing/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:38 UTC

SYNOPSIS

Your retrieval pipeline returns a document, the model cites it, and the claim checks out. Everything in the trace looks fine. But what if the model actually pulled that answer from its training data and the citation is just decoration? This episode digs into a genuine gap in AI observability: tracing shows what your agent did, never why. We walk through our own planning-agent-to-research-agent-to-scriptwriter pipeline as a case study, then dig into new mechanistic interpretability research on how LLMs decide to cite — and why current context-attribution tools can't separate what the model knew from what you handed it. If you're building knowledge-based generative products, this one reframes what your citations actually prove.

TRANSCRIPT



Corn

Our own pipeline is the case study today. Prompts come in, they go to a planning agent, the planner dispatches a research and grounding sub-agent that goes out with Exa, sometimes fetch, pulls findings off the open web, and those findings get injected into the scriptwriting agent before any of this reaches TTS.



Herman

Which is why we say "the sources" like it's a thing people should already know about.



Corn

Because it is a thing. It's a list. Daniel wrote in about exactly that. He wants to talk about tracing and observability as it applies to grounding, and he's coming at it from a very specific frustration. The grounding side of this pipeline has worked well for a long time. Every system has failure modes anyway. And when something inaccurate does land in an episode, the question of where it slipped in is hard to answer. It could be the model preferring its own stale parametric knowledge over something retrieval handed it. It could be the retrieval itself. And the model's reasoning about which one to trust isn't sitting in the logs.



Herman

Right, because the logs show the tool call, not the deliberation.



Corn

So to debug the occasional error you need visibility into which of several sources the model preferred, and why. Even if the fix is just a guardrail, always prefer this source over that one, or change the retrieval pattern entirely, you can't make that change without knowing where grounding actually pulled from. He wants to look at how observability can work in grounding, specifically for people building knowledge-based generative products, so they can iron out problems with retrieval, accuracy, and source citation.



Herman

There's a lot in there, and I think the honest framing is that this is a real gap, not a solved problem with bad documentation.



Corn

Then let's start with what tracing and observability actually are, and why this is getting urgent.



Herman

A trace, in the modern sense, is a waterfall of spans. Each span captures a tool name, input parameters, the response payload, latency, token count. That's the common data model across LangSmith, Arize Phoenix, Langfuse, Helicone, Braintrust. They all converged on it because it's the obvious shape for the problem.



Corn

And underneath it, OpenTelemetry.



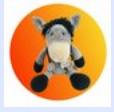
Herman

OpenTelemetry's GenAI semantic conventions, the gen_ai namespace, standardize LLM operations, agent structure, and tool calls as spans. So an agent is observable with the same tooling as the rest of your stack. OTel graduated from the CNCF in May, which cements OTLP as the common telemetry protocol. That part is settled now.



Corn

Settled enough that people assume it covers more than it does.



Herman

It covers what happened. You have complete visibility into what the agent did, and zero visibility into why. The planning layer, goal decomposition, hypothesis generation, constraint satisfaction, plan formation, all of that runs inside the LLM's forward pass, and in most production architectures its outputs are ephemeral. They don't get logged as



Herman (continued)

structured events because there's nothing clean to log.



Corn

So our planner to research to scriptwriter chain is exactly the multi-agent case where the tool traces show you the waterfall and nothing about the reasoning behind source selection.



Herman

Precisely the case. And I want to be careful here, because I think there's a version of this where people hear "the reasoning isn't logged" and think the fix is just more logging. It isn't. The reasoning isn't a discrete object you forgot to serialize.



Corn

It's a forward pass.



Herman

There's no span for it because there's no event. There's a computation.



Corn

So walk me through our pipeline properly, because I think people hear "we have a research agent" and picture something much tidier than what's actually there.



Herman

The planning agent receives the prompt. It decides what kind of episode this is, what the shape should be, what needs grounding. Then it dispatches the research and grounding sub-agent. That sub-agent goes out with Exa, sometimes with fetch when a specific page needs pulling, and comes back with a list of findings.



Corn

Which is literally a list. Not a document, not a corpus. A list of findings with some provenance attached.



Herman

And that list gets injected alongside the episode brief and the other inputs into the scriptwriting agent. So when we say "the sources," that's what we mean. It's the list of findings the research agent handed over, sitting in the context window next to everything else.



Corn

Here's the thing I keep circling. When an inaccuracy lands, the trace shows the research agent returned findings, and the scriptwriter produced text. Two clean spans. What it does not show is whether the scriptwriter used those findings or reached into its own parametric memory and produced something that happened to sound similar.



Herman

And those two things look identical from the outside.



Corn

They look identical in the output. They look identical in the trace. The only place they differ is inside a computation nobody logged.



Herman

This is where the citation research gets interesting, and I think it's the most useful thing to come out of the last year on this. There's a paper, "How Do LLMs Cite?", from van Dort and Heuss at the University of Amsterdam, presented at ECIR this year. It's the first mechanistic account of how a model decides to attach an inline citation.



Corn

Mechanistic meaning they went into the weights.



Herman

They went into the weights. They took Llama-3.1-8B-Instruct and asked: when the model produces a citation, what inside it made that happen? And the answer is not a citation head. There is no single circuit that does citing.



Corn

Of course there isn't.



Herman

It's a distributed, fragile ensemble. Attention heads and MLPs across depth, all contributing. They call it an attributional ensemble, and the word fragile is doing real work there. You can knock pieces out and citation behavior changes in ways that don't track with answer quality at all.



Corn

That's the part that should worry anyone building on this. Citation and correctness are separate systems.



Herman

Largely separate. And the mechanism they found is shallower than you'd want. The decision to cite relies heavily on entity co-reference matching. Not deep semantic engagement with the document. The model is matching entities, noticing that the thing it's talking about is also in the retrieved passage, and flagging a citation.



Corn

That's a heuristic.



Herman

It's a heuristic. An early MLP at layer zero enriches the queried entity, and that flag propagates to the question's second mention of the same entity. Patch that single activation and you can restore or break the citation even though the question text is identical.



Corn

So the citation decision can flip without the content changing at all.



Herman

Without a single word changing. That's the demonstration.



Corn

Say the number on the intervention.



Herman

Amplifying the pro-citation components recovered more than ninety percent of missed citations, with negligible impact on answer accuracy. Down-scaling the necessary components suppressed sixty-nine percent of spurious citations. Nine out of thirteen.



Corn

Spurious meaning the model was citing something it shouldn't have.



Herman

Citing something that didn't earn the citation. And on HotpotQA, baseline strict correct citation was zero point zero zero one. At alpha one point two that rose more than twenty-fold, to zero point zero two four.



Corn

Zero point zero zero one to zero point zero two four. That's the range we're operating in.



Herman

That's the range. The absolute numbers are grim. The relative improvement is real. Both of those things are true and I think you have to hold them together.



Corn

Here's what I want to sit with, because I think it's the heart of Daniel's question. The paper draws a distinction between citation correctness and citation faithfulness. Correctness is: does the source factually support the claim? Faithfulness is: did the model actually use the source to generate the claim?



Herman

And those come apart constantly. A model can recall an answer from training data and attach a citation to a document that merely confirms it. The citation is factually right. The document does support the claim. And the document had nothing to do with producing it.



Corn

Correct but unfaithful.



Herman

That's the failure pattern, and it's exactly the one you can't see in a trace, because everything in the trace looks fine. Retrieval returned a relevant document. The output cites it. The claim is true.



Corn

And the model's actual reason for producing the claim was that it memorized it in training.



Herman

Which means your grounding layer is doing nothing in that instance, and you have no signal telling you so. The paper's own phrasing is that the citation can be generated by a separate heuristic process running in parallel to answer generation, and that LLM outputs can create a misleading narrative of their own reasoning. Their conclusion is blunt. If citations are merely cosmetic, appended after the fact rather than driving the answer, then RAG systems risk creating a false sense of security.



Corn

False sense of security is the phrase I'd underline. For us specifically.



Herman

For us specifically, yes. We tell listeners there are sources. There are sources. They exist, they were retrieved, they're in the context. Whether they drove any particular sentence is a different question and we do not know.



Corn

I don't think most people building on retrieval want to hear that.



Herman

I don't think most people building on retrieval have asked it.



Corn

So the entity co-reference finding, the layer-zero MLP, the fact that patching one activation flips the citation. Why does that matter for debugging rather than just being a neat interpretability result?



Herman

Because it tells you the citation is not downstream of the reasoning. It's running alongside it, on a shallower signal. So when you look at a trace and see that the model cited a source, you are not seeing evidence that the source informed the answer. You're seeing



Herman (continued)

evidence that an entity matched.



Corn

And entity matching is exactly what a retrieval system is optimized to produce.



Herman

Exactly what it's optimized to produce. Which means the citation signal and the retrieval signal are correlated by construction, and the correlation tells you nothing about causation.



Corn

That's a clean way to put it. The thing you're measuring and the thing you care about are correlated for reasons that have nothing to do with what you care about.



Herman

Which brings the whole debugging problem into focus. You see a citation. You cannot infer faithfulness from it. You cannot infer it from the trace. And the mechanism generating it is a heuristic that fires on surface features.



Corn

So where does that leave the remediation Daniel's asking about? He says even if the fix is just a guardrail, always prefer a certain source, you need to know where grounding pulled from. Does the research support that being tractable?



Herman

This is where it gets worse before it gets better. There's a paper from Aalto, Trinh, Zhu, and Szyller, called "How Context Attribution Handles What the Model Already Knows." They took the four mainstream context attribution methods. ContextCite, AttriBot, TracLLM, TokenShapley.



Corn

These are the tools you'd reach for.



Herman

And they tested them on the exact question we care about: when the context overlaps with what's in the training data, can the method tell you which one the model actually used?



Corn

And?



Herman

It cannot. Source separation precision stayed near chance for all four. Chance is zero point five. The best was TokenShapley at zero point five two on LLaMA3-8B.



Corn

That's a coin flip with a slight lean.



Herman

It's a coin flip with a slight lean. And the concrete failure is beautiful in a grim way. A supporting document span can receive a near-zero attribution score, because removing it doesn't change the output. The model already knew it.



Corn

So the method penalizes the document for being redundant with the weights.



Herman

It penalizes the document for being redundant with the weights. And you cannot tell from the score whether you're looking at an irrelevant document or a document whose knowledge is already in the model. The authors say it directly. A low score means the segment is irrelevant, or that its knowledge is already in the weights. Those are the same number.



Corn

Which kills the guardrail.



Herman

Which kills the naive guardrail. If your rule is "discard sources with low attribution scores," you will discard exactly the sources the model already agrees with, which are the ones you'd most want to keep if you're trying to anchor the model to retrieved reality rather than its own memory.



Corn

That's the remediation paradox. The signal you'd use to decide whether to trust a source is contaminated by the very thing you're trying to detect.



Herman

Their conclusion is that the attribution score is a property of the context-model pair, not the context alone. And that likelihood-based metrics, Drop@k, LDS, tend to favor methods whose scoring mechanisms match their own perturbation assumptions. So the benchmarks are partly measuring self-consistency.



Corn

Which is a polite way of saying the evaluation is rigged toward the thing being evaluated.



Herman

It's a real methodological problem. I don't think it's malicious. But yes.



Corn

Now bring in the multi-turn case, because our pipeline is multi-turn whether we think of it that way or not.



Herman

Microsoft's Tokengeist, from June. It introduces multi-turn context attribution, and the headline finding is that flat attribution methods achieve under twenty percent source recall on multi-hop dependencies. Tokengeist reaches ninety.



Corn

Under twenty to ninety is a chasm.



Herman

It's a chasm, and the reason is that flat methods attribute to a single turn or a single span, and the actual provenance runs across turns. They named the failure pattern provenance collapse. The benchmark is MTCABench, three thousand eight hundred and forty-five target spans across six hundred and sixty-five multi-turn conversations, with gold provenance graphs up to depth fourteen.



Corn

Depth fourteen.



Herman

So the chain of reasoning that produced a given token can run fourteen steps back through the conversation. Flat attribution sees the last step and calls it the source.



Corn

Our planner hands the research agent a constraint. The research agent hands the scriptwriter findings. The scriptwriter produces a sentence. If the sentence is wrong because the planner's constraint was stale, no attribution method looking at the scriptwriter's context will find it.



Herman

None of them will find it. The provenance runs back through a handoff that isn't in the context window at all.



Corn

That's the case Daniel's describing without naming it.



Herman

It's exactly that case. And it's why the DSG work is interesting, even though it's solving a slightly different problem.



Corn

Decoupled search from reasoning.



Herman

The argument in DSG is that native search grounding bundles retrieval policy, provider choice, evidence injection, cost, latency, and generation behavior behind a single model-provider boundary. So grounding is hard to inspect, hard to tune, hard to reuse, hard to port. You can't see inside it and you can't swap pieces.



Corn

Because the provider owns the whole stack.



Herman

The provider owns the whole stack. What DSG does is move grounding outside the reasoning model, into an MCP-compatible gateway that exposes provider routing, source-aware context rendering, and retrieval-depth control as first-class controls.



Corn

First-class meaning you can log them, version them, change them independently.



Herman

First-class meaning they're part of the interface rather than buried in the provider's implementation. And the numbers are good. SimpleQA, eighty-six point one percent accuracy versus native eighty-seven point seven, at ninety-one percent lower search cost. Warm-cache hit rate ninety-nine point four percent, sixty-eight percent lower latency. On an e-commerce workload they cut search cost by more than ninety-eight percent.



Corn

So you give up one and a half points of accuracy and get an inspectable boundary.



Herman

You give up one and a half points of accuracy and get an inspectable boundary, and you get the cost and latency wins as a side effect. The paper's framing is that real-time grounding is best treated as an optimizable interface boundary, not a fixed model feature.



Corn

That's the architectural answer. Does it solve the faithfulness problem?



Herman

No. It makes grounding legible. It doesn't make attribution faithful. You still can't tell from the trace whether the model used the retrieved evidence or its own memory. You've just moved the boundary to somewhere you can see it.



Corn

Which is progress. It's just not the thing Daniel's asking for.



Herman

It's not the thing Daniel's asking for. And I want to be honest about the state of things here, because I think the honest answer is the useful one. I could not find an off-the-shelf tool that directly answers "did the model use the retrieved source or its parametric memory." The closest things are research prototypes. ContextCite, TracLLM, AttriBot, TokenShapley, Tokengeist, MIRAGE. And the Aalto work shows all four mainstream methods fail at exactly this disentanglement.



Corn

So the capability exists in papers and not in products.



Herman

In papers and not in products. And there's a second gap that's less discussed. There is no standard OpenTelemetry semantic convention for grounding-source attribution. No gen_ai attribute for "which retrieved document drove this token." Teams are defining their own attribute namespaces, which means nothing is portable and nothing is comparable across vendors.



Corn

So even the plumbing is bespoke.



Herman

Even the plumbing is bespoke. You can trace a tool call with a standard span. You cannot trace a source commitment with a standard span, because there isn't one.



Corn

Now I want to get to the part of Daniel's question that's uncomfortable for us specifically. He says the internal reasoning isn't always exposed in logging and traces. Why is that? Is it a technical limit or a choice?



Herman

Both, and increasingly the second one. OpenAI treats hidden chain-of-thought as a monitoring object rather than something user-facing. Gemini exposes thought summaries rather than raw thoughts. Claude's extended thinking gives you controlled transparency, which is a careful phrase.



Corn

Controlled by whom.



Herman

Controlled by the provider. And there's a reason for that beyond competitive secrecy. Anthropic's work, "Reasoning Models Don't Always Say What They Think," found that models given hints that changed their answers verbalized those hints far less often than they acted on them.



Corn

So the model acts on the hint and doesn't mention it.



Herman

Acts on it and doesn't mention it. Which means the reasoning trace is an imperfect oversight channel. It's not that the model is lying. It's that the verbalization and the computation are not the same process.



Corn

That's a much stranger claim than "the model hides things."



Herman

It's a much stranger claim. And there's a July preprint, revised earlier this month, that makes it stranger. Frontier models do part of their work in what the authors call filler tokens carrying no interpretable content. And those tokens buy accuracy gains up to thirteen percentage points across thirteen models.



Corn

Filler tokens that mean nothing and improve accuracy.



Herman

Tokens that carry no interpretable content and improve accuracy by up to thirteen points. Which means part of the computation is happening in a channel that has no readable surface at all. The conclusion people are drawing is that a reasoning transcript is an incomplete trace, not an audit log.



Corn

Incomplete trace is generous. It's a summary of a summary.



Herman

It's a summary of a summary, and the underlying thing it's summarizing isn't fully legible either.



Corn

So stack the three problems. The trace shows what happened, not why. The reasoning transcript is incomplete even when you have it. And the attribution methods that would tell you which source won are at chance when the context overlaps with training data. Daniel's question is sitting on top of three separate gaps.



Herman

Three separate gaps, and they compound. If the model preferred its parametric memory, the trace won't show it, the reasoning won't mention it, and the attribution method will give you a score you can't interpret.



Corn

And the output will have a citation on it.



Herman

And the output will have a citation on it, because the entity matched.



Corn

I want to go back to the guardrail idea, because I think it's where most builders will actually try to intervene. Daniel says the remediation might be as simple as always prefer a certain source, or change the retrieval pattern. Is that workable given what the Aalto work shows?



Herman

Changing the retrieval pattern is workable. Always prefer a certain source is workable if the preference is defined on the source rather than on the model's response to it. Where it breaks is when you try to use attribution scores as the trigger.



Corn

Because the score is a property of the pair.



Herman

If you want a guardrail, build it on something exogenous. Source tier, recency, domain, whether the document was fetched versus recalled from cache. Those are properties of the source. Attribution scores are properties of the interaction, and they move when the model's weights move.



Corn

Which means your guardrail drifts when you upgrade the model.



Herman

Your guardrail drifts when you upgrade the model, and you won't notice, because the scores still look like scores.



Corn

That's the failure pattern I'd worry about most. Not that the guardrail doesn't work. That it works, silently stops working, and keeps reporting that it works.



Herman

That's the one. And I don't have a clean answer for it. I think the honest position is that attribution is not yet a production-grade control surface. It's a diagnostic you use with a human in the loop, and you should treat its output as a hint rather than a signal.



Corn

So what does a builder actually do this quarter?



Herman

Log the retrieval. Log what was returned, from where, with what provenance. Make grounding an explicit boundary in your architecture, the way DSG argues, so you can see and change it. And accept that the faithfulness question is currently unanswerable and design around that, rather than pretending a citation means what it looks like it means.



Corn

Which is a worse answer than Daniel was hoping for and probably the true one.



Hilbert

Can I ask you something about the trace itself.



Corn

Go ahead.



Hilbert

When you say the trace shows what happened and not why. I spent a couple of years as a quality inspector at a commercial printing plant, and the press had logs. Every station reported. You could see which station ran, when it ran, how long it ran, what the temperature was. And when a defect came off the end of the line, the logs told you nothing about which station caused it. You had to infer it from the paper.



Herman

From the paper itself.



Hilbert

From the paper. You'd look at the registration drift, the ink lay, the way the fold sat, and you'd work backwards. The logs were a record of activity. They weren't a record of cause. Same thing you're describing, I think.



Corn

That's exactly the shape of it.



Hilbert

What I'd say, though, is that the operators who solved the most defects weren't the ones with better logs. We had two guys who were very good at reading logs. They were fine. The person who solved the most was a woman named Ruth who walked the line every morning before the shift started. Same route, every day, forty minutes. She knew which station ran warm, which one had a roller that needed replacing before it failed, which one drifted when the humidity changed. None of that was in the logs. It was in her.



Herman

Tacit knowledge of the specific machine.



Hilbert

Tacit knowledge of that specific machine. And I don't know if an AI pipeline has an equivalent of walking the line. I'm not sure it can. There's no physical thing to walk.



Corn

There's no morning route.



Hilbert

But there's a second thing we did that might be closer to useful. After a while we started deliberately introducing small defects. A tiny misregistration at a known station, a slight ink variation. Harmless. We'd throw it in on purpose, at a station we'd picked, and then watch what came off the end and compare it to the logs. We were calibrating. We wanted to know what the trace looked like when we already knew the answer.



Herman

Known-answer probes.



Hilbert

I don't know what you'd call it. We called it Tuesday. But the point was that you can't interpret a trace you've never seen the other side of. You need cases where you know the cause, so you know what the symptom looks like.



Corn

So you build a library of known failures.



Hilbert

You build a library of known failures. And then when something new comes off the line you've got something to compare it to. I don't know if that translates. It seems like it might.



Corn

It translates more than you're giving it credit for.



Hilbert

Anyway. I have a dentist appointment in Tel Aviv and I'm already late.



Corn

Go.



Herman

The known-answer probe idea is the thing I hadn't thought about. We've been talking about better instrumentation, and what Hilbert's describing is calibration. You deliberately break the pipeline in a controlled way, at a known stage, and you look at what the trace shows. Then you know what a retrieval failure looks like versus a parametric-memory failure versus a planner failure.



Corn

Because right now we have no idea what those look like. We've never seen one where we knew the answer.



Herman

And you can construct them. Feed the pipeline a question whose answer you've deliberately corrupted in the retrieval layer, and see whether the output follows the corrupted source or the model's memory. That's a faithfulness test.



Corn

That's a faithfulness test that doesn't require any of the attribution methods.



Herman

It doesn't require any of the attribution methods. It requires you to control the input, which you can do, and to look at the output, which you already do. The thing you can't do is see inside. So you stop trying to see inside and you infer from controlled perturbation.



Corn

Which is what Ruth was doing, in a way. She knew the machine well enough to know what normal looked like.



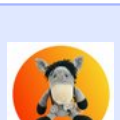
Herman

She knew what normal looked like. And the synthetic defect was how you taught everyone else.



Corn

So the practical answer for builders is: instrument what you can, make grounding an explicit boundary, and then build a calibration set of known failures, because the trace will never tell you why on its own.



Herman

And be honest that the citation is not evidence of use. That's the thing I'd want people to take away. A citation means an entity matched. It doesn't mean the source drove the answer.



Corn

The cutting-room floor item, since we're near the end. There's a paper on the attribution-compression frontier that quantifies how context compression breaks citation attribution. Which matters because everyone compresses context to save tokens.



Herman

And compression is exactly the operation that destroys the surface features the citation heuristic fires on. You compress, the entity match disappears, and the citation behavior changes in ways nobody's tracking.



Corn

Worth a whole episode. So where does this leave us. If attribution methods can't disentangle in-context from in-weight contributions, what would it take to build that capability? Better mechanistic interpretability, better architecture like DSG's decoupled grounding, or something else entirely.



Herman

My honest guess is that it's the architectural path first, because it's tractable now, and the interpretability path second, because it's the one that would actually answer the question. Making grounding an inspectable boundary gets you legibility. It doesn't get you faithfulness. And as agentic pipelines become more common in knowledge-based products, the gap between what traces show and what builders need to know is going to widen, not narrow.



Corn

Unless grounding becomes a first-class observability concern rather than a feature you buy from a provider.



Herman

Unless that. Which is a choice somebody has to make.



Corn

Thanks to our producer, Hilbert Flumingtop.



Herman

This has been My Weird Prompts.



Corn

If you're getting something out of these, a review helps other people find the show. We'll be back soon.