Episode Book

EPISODE #5481

Small Models, Big Guardrails: PII Detection in ChatGPT

PUBLISHED	23 September 2026 at 11:32 UTC
DURATION	25:54
LISTEN	myweirdprompts.com/episode/small-models-pii-detection/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.2
BOOK GENERATED	19 October 2026 at 17:36 UTC

SYNOPSIS

When ChatGPT asks you to approve a tool call, something has already scanned the outgoing data for usernames, addresses, and other PII — and that something is probably a small language model. This episode digs into the "guardian" class of compact models doing narrow, high-volume security work: PII-Tracer, OpenAI Privacy Filter, GLiNER2-PII, and the consistency problem that makes partial detection worse than it looks. We also untangle a name that doesn't exist, and explain why character-level boundaries matter more than parameter counts.

TRANSCRIPT



Corn

Daniel's been poking at the ChatGPT connectors again, and he's noticed something in the approval modal that he thinks is a small model doing security work. Herman Poppleberry, say hello.



Herman

Hello. And he's probably right, which is the fun part.



Corn

Here's what he wrote. He says we've talked about small language models before, the unglamorous accessory models that do integral work while the big reasoning models get all the attention. He's been working at the MCP layer for a while, so he knows how the plumbing goes. What he's noticing is this: in ChatGPT, when the model goes to invoke a tool call, a modal appears in the chat window. Approve or deny. But the modal also carries flags. It'll say something like contains username, contains address. So before the MCP server is ever touched, something has scanned the outgoing data, identified the PII, and the user is being asked to approve or deny on the basis of that scan. His read is that we're watching smaller models for PII detection and security in action. A specialized subset that lives inside the chat thread and assesses sensitivity before anything leaves. He mentions a few of these are open source on Hugging Face, including one he calls PII Guard by Perplexity. And he wants to know how this class of model works under the hood, and how developers building their own systems can wire it in.



Herman

There's a correction sitting in the middle of that, and it's a good one, so let's get to it.



Corn

Straight to the correction. No warm-up.



Herman

PII Guard by Perplexity doesn't exist under that name. Perplexity's actual artifact is pplx-pii-masking, which they also call PII-Tracer. It shipped on the first of September with a benchmark called PII-TRACE. PII Guard is a separate thing entirely, a regex-based guard sitting in the presidio-oss hai-guardrails repo. Two real things, one invented name, and Daniel's folded them together.



Corn

That's a very Daniel mistake to make, and I mean that fondly, because he's usually the one catching everyone else's naming drift.



Herman

It's also the most instructive part of the prompt. The name propagated because it sounds right. PII Guard, by Perplexity. Of course that's what it's called. Except it isn't, and if you go looking for it on Hugging Face you'll find the embed collections and the pii-masking family and nothing called PII Guard.



Corn

So before we get into the modal, let's do the definition work. Small language models.



Herman

Compact models. Roughly a tenth of a billion parameters up to a few billion. They get deployed for narrow, high-volume jobs: classification, routing, guardrails, PII detection. The reason they matter isn't that they're clever. It's that they run on-device or on cheap CPU. No server round-trip. Privacy, latency, and cost all fall out of that one property.



Corn

And the literature has a name for them in this role.



Herman

Guardians. There's an EMNLP industry paper from 2024, SLM as Guardian, arguing a smaller LLM can handle both harmful-query detection and the safeguard response generation, and that this avoids the training cost and the helpfulness degradation you get when you bake safeguards into a large model. There's a follow-on, LiteLMGuard, on lightweight on-device guardrails. And this year there's work evaluating twenty-two compact open-weight models on routing plus out-of-distribution rejection under latency constraints. So this is a deployment pattern people are actively measuring, not a thought experiment.



Corn

Now the modal.



Herman

The modal is the hook because it's visible. You're in a chat, ChatGPT decides it needs to send an email through the Gmail connector, and up comes the box. Approve. Deny. And underneath, those flags naming what it found. Username. Address. The thing to understand is that this is a distinct control surface from the sandbox.



Corn

Meaning what, exactly?



Herman

Meaning the network proxy that governs what the agent can reach does not filter web search, app or connector tool calls, MCP server connections, browser activity, Computer Use. Those run through separate service connections with separate policies. So the approval modal isn't a side effect of the sandbox. It's its own gate.



Corn

And what does OpenAI actually document about when approval is required?



Herman

Their agent docs say destructive app or MCP tool calls always require approval when the tool advertises a destructive annotation, unless the tool advertises a read annotation, which takes priority. So the tool itself declares what it is, and the harness decides whether to interrupt you.



Corn

Which brings us to the question Daniel's actually asking. Is a small specialized model generating those flags?



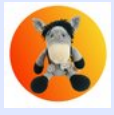
Herman

Honest answer: OpenAI has not said. I've looked, and there's no source stating that the PII flags in the ChatGPT approval modal come from a small language model. That's an inference. A plausible one, and I'll show you why it's plausible, but an inference.



Corn

Give me the plausible version.



Herman

Perplexity documented the exact architecture. Their blog describes PII-Tracer as providing one local control signal for model routing by flagging spans predicted to contain PII, and then the application enforces the routing policy. It keeps the input local, redacts the detected spans, or requests explicit approval before escalating to a cloud model. Read that last clause again. Requests explicit approval before escalating. That is functionally the ChatGPT modal pattern, written down by a different company.



Corn

So the shape matches. The attribution doesn't.



Herman

Right. And there's a second candidate. OpenAI ships something called an automatic approval reviewer. You can set approvals_reviewer to auto_review, and then a reviewer agent evaluates eligible approval requests before they run. It checks for data exfiltration, credential probing, persistent security weakening, and destructive actions. Low and medium risk proceed, critical risk gets denied.



Corn

That's a model-based gate.



Herman

It is. But it's an agent making extra model calls, not a PII detector. The docs even warn that auto-review uses extra model calls, so it adds to your usage. And nothing in it is described as scanning for a username or an address.



Corn

So the modal could be three different things and we don't know which.



Herman

It could be a small specialized model. It could be a regex and Presidio-style scanner, which is what OpenAI's own Guardrails library uses for its Contains PII check. Or it could be the auto-review agent. Or some combination. The prompt's hypothesis is the most interesting of the three, and it matches Perplexity's documented design, but nobody at OpenAI has confirmed it.



Corn

Fine. Let's take the hypothesis seriously on its own terms, because the model Daniel's gesturing at is real and it's worth understanding.



Herman

PII-Tracer. Six hundred million parameters. Bidirectional Qwen3 encoder, backbone is pplx-embed-v1-0.6b with bidirectional attention turned on. Two heads on top. The first is a thirty-seven class BIOES token-classification head over nine PII categories. Private person, account number, private URL, private date, private address, private email, private phone, other PII, and secret. That head decodes through a constrained Viterbi decoder, which matters because it stops the model emitting structurally impossible tag sequences. The second head is conversation-level, a sensitivity score for the whole thread.



Corn

Four thousand token window?



Herman

Four thousand ninety-six. Trained three epochs on about seven hundred fourteen thousand samples. MIT licensed.



Corn

Now the comparison, because this is where it gets interesting.



Herman

OpenAI Privacy Filter. One and a half billion parameters. It takes an autoregressive language model and converts it into a bidirectional PII detector. Thirty-three entity types, evaluated across thirty-two benchmarks, fourteen languages, five domains. Zero-shot F1 of 0.855 on AI4Privacy. And then 0.464 on the SPY medical set.



Corn

That's a big drop.



Herman

It gets worse out of distribution. On general NER it lands somewhere between 0.04 and 0.40. It collapses on Arabic at 0.04 and Cyrillic at 0.03. It's recall-biased, precision between 0.31 and 0.54, recall 0.70 to 0.85. Strong on email at 0.78 and phone at 0.76. Weak on person names at 0.40 and addresses at 0.49.



Corn

So the bigger model is worse.



Herman

On the specific job, yes, and the character-level numbers make it starker. PII-Tracer's character F1 is 0.629, highest among twelve systems evaluated. GPT-5.6-sol and Claude Sonnet 5 are comparable on overall span F1 but lower on character F1. And there's GLiNER2-PII, three hundred million parameters, forty-two entity types, trained on a multilingual synthetic corpus of four thousand nine hundred ten texts, which takes the highest span-level F1 among five systems on the SPY benchmark.



Corn

Three hundred million parameters beating a one and a half billion parameter model.



Herman

On that benchmark, yes. Which tells you the parameter count isn't the story. The training distribution is the story. These are narrow models trained on the exact shape of the problem.



Corn

Hold on. Say the character F1 thing again, because I want to be sure I've got it.



Herman

Character F1 measures whether the model got the boundaries right, not just whether it found something PII-shaped. If it flags "Daniel Rosehill" as a person but clips the surname, span-level scoring might forgive it. Character-level scoring won't. And character boundaries are what matter when you're deciding which bytes to redact.



Corn

So a model that finds the PII and a model that finds the PII precisely are different things, and the second one is the one you can actually act on.



Herman

That's the whole game.



Corn

Let me push on the part I think is underrated. Finding most of the PII is not the same as finding every copy of it.



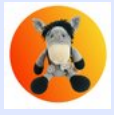
Herman

That's the consistency problem, and it's the sharpest result in the whole stack. PII-Tracer finds every mention of 79.4 percent of recurring identifiers, and 77.6 percent of cross-turn identifiers. GPT-5.6-sol manages 57.0 and 55.1.



Corn

And when the identifier shows up a lot?



Herman

Six to ten mentions, PII-Tracer scores 0.691. GPT-5.6-sol drops to 0.464. GLiNER2-PII at 0.073. Claude Opus 4.8 at 0.045.



Corn

Four and a half percent.



Herman

On that specific task, yes. And here's the shape of the decay. PII-Tracer goes from 91.7 percent at a single mention down to 69.1 percent at six to ten mentions. The competitors don't decay, they fall off a cliff, down to somewhere between 4.5 and 7.3 percent.



Corn

So the small model is the only one still standing, and it's still missing one in three.



Herman

That's the point. Perplexity's own line is that one missed mention is enough for personal information to pass through. If a name appears in the greeting, the body, and a signature block, and you catch two of the three, you haven't redacted the document. You've redacted most of it.



Corn

Which is a strange failure mode, because it looks like success. The flag says contains username. The user sees a flag, feels protected, clicks approve.



Herman

And the third mention goes out.



Corn

There's a fix for this, isn't there? Something that doesn't require retraining.



Herman

Sliding-window decoding with fifty percent overlap. You run the detector over overlapping chunks instead of one pass, and you reconcile. Recall goes from 0.830 to 0.965. Multi-mention consistency from 0.794 to 0.954. No retraining, no new weights, just a different way of feeding it.



Corn

That's a deployment detail that costs you compute and buys you the thing that actually matters.



Herman

And it's free to anyone who reads the model card.



Corn

Now the philosophy question, because there are two gates here and they're built on different beliefs.



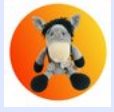
Herman

Perplexity's gate is a dedicated encoder. Six hundred million parameters, purpose-built, fast, local, deterministic in the sense that it always runs the same way. OpenAI's auto-review is an agent. It reasons about intent. It can look at a tool call and ask whether this looks like exfiltration rather than whether this string looks like an email address.



Corn

And the cost.



Herman

The docs say it plainly. Auto-review uses extra model calls, so it can add to your usage. You're paying in latency and tokens for the reasoning.



Corn

The encoder is cheap and blind to intent. The agent is expensive and understands what you're trying to do.



Herman

And there's a live argument underneath that about what actually works at runtime. Classifier-based guardrails versus LLM-driven guardrails. OpenAI's own Guardrails library ships a Contains PII check that's built on Microsoft Presidio in Python and a built-in regex engine in TypeScript. Regex. Meanwhile Perplexity and GLiNER2 are arguing for learned detectors.



Corn

Regex is fast, auditable, and completely unable to tell you that "the guy who lives above Hannah" is a person reference.



Herman

It's also unable to be wrong in a way you can't explain. Every regex match is a rule you wrote. Every model output is a probability.



Corn

Which one would you ship?



Herman

Both, in sequence. Regex first because it's free and catches the structured stuff, the account numbers, the card numbers, the things with a fixed shape. Then the learned model for everything the regex can't see. That's not a compromise, that's just what the two tools are each good at.



Corn

Okay. So somebody's building a pipeline. What do they actually wire in?



Herman

Depends where the data flows. If you're using LangChain, there's a PIIMiddleware for agents that detects emails, credit cards, IPs, MAC addresses, URLs, on both input and output, and it ships a stream transformer so every wire surface is covered. That last part matters more than the detection list. Most people remember to check the prompt and forget the response.



Corn

And if you're not in LangChain?



Herman

AWS publishes a Claude Code plugin hook, pii_prompt_guard.sh, that scans the UserPromptSubmit and PreToolUse events and exits with code 2 to block when it finds PII or secrets. Code 2 is the block signal. That's a shell script, not a model, but it's the same gate concept.



Corn

There's a community one too.



Herman

agent-seatbelt, a project called hooks-opf. It describes itself as a userland PII detector for AI coding agents, catching secrets and personal data flowing into the prompt or out of tool responses before the LLM ever sees the bytes.



Corn

Before the LLM ever sees the bytes. That's the design principle in six words.



Herman

And it's the right one. Once the bytes are in the context window, you've already lost. Everything downstream is mitigation.



Corn

Then there's the deployment packaging, which I think is the most underrated part of this whole story.



Herman

Perplexity shipped three variants. pii-masking-vllm, servable through vLLM. pplx-pii-masking-MLX, int8, seven hundred seventy-nine megabytes, sixty-five percent of the bf16 size, running on Apple Silicon on-device. And lemonade-sdk's pplx-pii-masking-onnx, CPU through onnxruntime.



Corn

Seven hundred seventy-nine megabytes on a laptop.



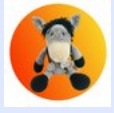
Herman

And they call it lossless. So the privacy gate runs on the machine, next to the data, and nothing has to leave to be checked.



Corn

Which is the whole argument. A frontier model can beat PII-Tracer on span-level F1 and it still loses, because it can't run locally. If your privacy gate requires sending the data to a server to find out whether the data is safe to send to a server, you haven't built a gate. You've built a detour.



Herman

That's the thesis in one data point, and it's why the small model wins even when it isn't the best model.



Corn

Let me try the other side of it. Is there a case where you'd want the big model doing the PII detection?



Herman

If you're processing at low volume and the data is already in a trusted environment, sure. If the sensitivity judgment requires understanding intent, sure. GPT-5.6-sol does beat PII-Tracer on span-level F1. But the moment the requirement is that the data never leaves, the frontier model is disqualified. Not outscored. Disqualified.



Corn

Different competition.



Herman

Different competition entirely.



Corn

I want to go back to the naming thing for a second, because I think it's more than a trivia correction.



Herman

It's a symptom. Model names propagate through the ecosystem by sound, not by source. Somebody says PII Guard, somebody else says by Perplexity, and now there's a model that doesn't exist that people are searching for on Hugging Face. And the practical cost is real. If you're a developer trying to find the right tool and you search the wrong name, you either find nothing or you find the regex guard and think it's the encoder.



Corn

And those two things behave completely differently.



Herman

Completely. One is a set of rules. The other is a six hundred million parameter model with a Viterbi decoder. Same problem, same vocabulary, nothing else in common.



Corn

Which raises the question of how you'd even evaluate one against the other.



Herman

You'd use a benchmark built for the task. PII-TRACE is thirteen thousand one hundred forty-eight synthetic conversations, thirteen languages, ten writing systems, thirty-seven thousand four hundred thirty-one identifier mentions, nine PII types, and forty-one percent of them contain structured content. That last number is deliberate. It's making sure the benchmark doesn't just reward regex-shaped wins.



Corn

And on the external benchmarks?



Herman

PII-Tracer beats OpenAI Privacy Filter on all five. ai4privacy 0.950 against 0.907. Nemotron-PII 0.847 against 0.709. SPY 0.585 against 0.543. Gretel 0.952 against 0.895. TAB 0.594 against 0.350.



Corn

That TAB gap is enormous.



Herman

It is. And I don't know exactly what's driving it, honestly. I'd want to read the benchmark construction before I made a strong claim about it.



Corn

Fair. Let's pull back to the modal one more time, because I want to be precise about what we do and don't know.



Herman

What we know: the modal exists. It shows approve and deny. It shows PII flags. Approval is required for destructive tool calls unless the tool declares itself read-only. The network proxy doesn't cover connector calls, so the modal is its own gate. OpenAI ships an auto-review agent that checks for exfiltration, credential probing, persistent security weakening, and destructive actions, and denies critical risk.



Corn

What we don't know.



Herman

Whether a small specialized model generates those flags. Whether it's Presidio, whether it's regex, whether it's the auto-review agent, or whether it's something else entirely that isn't documented. Daniel's hypothesis is the most interesting reading and it matches a documented architecture from a different company. That's as far as the evidence goes.



Corn

I'd rather have it that way than have him be confidently wrong.



Herman

Same. The interesting version of this story is the one where we can say the pattern is real and the attribution is unconfirmed.



Corn

There's a thing I keep circling, though. Every one of these systems assumes a human reads the flag.



Herman

That's the assumption I'd bet against.



Corn

You'd bet against it hard.



Herman

I've watched people click through consent dialogs for thirty years. The modal is designed to communicate risk. It is not designed to make you stop.



Corn

Which is where the show's producer usually has something to say.



Hilbert

I spent about eight months in a room with a marker.



Corn

Go on.



Hilbert

Firm handled medical records. Before anything left the building it had to be redacted. Names, addresses, dates of birth. You'd get a stack in the morning and you'd work through it. Black marker, page by page. You got fast at it. That was the problem.



Herman

Fast how?



Hilbert

You stop reading. You start scanning for the shape of a name. Capital letter, capital letter, and your hand moves. By the third week I could do a hundred pages before lunch and I couldn't have told you what any of them were about. And the ones that got through were never the obvious ones. Nobody misses the name at the top of the intake form. You miss the name in the footer. You miss it in the header on page two. You miss it in a footnote where it's been abbreviated to initials and a surname, and your eye goes right past it because it doesn't look like the thing you're hunting for.



Corn

So the failure isn't attention. It's pattern-matching.



Hilbert

The pattern-matching is what makes you fast and it's what makes you miss. Same thing. And the modal's the same shape. It tells you it found a username and an address, and it puts two buttons underneath. It's built so you feel covered. It isn't built so you check.



Herman

That's the consistency problem in human form.



Hilbert

There was one. A social security number in the margin of a page I'd already signed off on. Somebody had written it in by hand next to a note. I found it on a re-read, weeks later, going through the same box for something else. I never told anyone.



Corn

Why not?



Hilbert

Because it had already gone. Telling them would have been a report about a thing that was already out the door. I put the page back in the box.



Corn

And the box is still there.



Hilbert

The box is still there. Anyway, I've got a thing at four.



Herman

The marker's the thing I'm stuck on. Eight months of a human being the PII detector, and the failure pattern is exactly the failure pattern we've been describing. It finds most of it. It never finds all of it. And the reason it doesn't is that it got good at the job.



Corn

The competence is the vulnerability.



Herman

And the model has the same shape. Ninety-one point seven percent at one mention. Sixty-nine point one at six to ten. It's not that it's bad. It's that it's confident, and the confidence is what you'd build a workflow on top of.



Corn

Here's the misconception I want to nail down, because I think it's the one most people carry.



Herman

Go.



Corn

The belief is that the approval modal means the data has been checked. That a flag saying contains address is a guarantee that every address has been found. And that's not what any of these systems do. They find most of it, they report what they found, and the reporting is what creates the feeling of safety.



Herman

The flag is a detection result, not a clearance. Nothing in the modal says the scan was exhaustive. It says here's what I found. Those are different claims, and only one of them is true.



Corn

So the correction is: the modal tells you what was caught, not what was missed.



Herman

And the gap between those two is where every leak lives.



Corn

One thing I'd want to leave people with. OpenAI hasn't said whether the modal's flags come from a small model, a regex scanner, or the auto-review agent. That's still open.



Herman

And the naming gap is its own small story. PII Guard by Perplexity is two real things wearing one wrong name. Perplexity's actual artifact is pplx-pii-masking, PII-Tracer, shipped with the PII-TRACE benchmark on the first of September. PII Guard is a regex guard in a different repo entirely. When developers go looking, that matters.



Corn

The privacy gate is becoming the most important model in the stack, and it's the one nobody talks about. The reasoning model gets the headlines. The six hundred million parameter encoder decides what the reasoning model is allowed to see.



Herman

And it's running on your laptop.



Corn

If you've got thoughts on any of this, a review helps other people find the show. Thanks to Hilbert Flumingtop for producing.



Herman

This has been My Weird Prompts.



Corn

We'll be back soon.