When the Harness Carries the Model

2026-06-06 / 00:17:31

> *“The benchmark measures the model; the repair layer is what you actually ship.”*
>
> — from this episode's transcript

- Lenar Kess
- Damra Vol

An open-weights model that fumbles tool calls on its own can go toe to toe with a frontier closed model — once you wrap the right error-handling around it. That gap, between what a model scores and what it does inside your repo, runs through everything we covered today.

- Ahmad Awais on Latent Space describes "tool confusion" — open models repeating the same invalid tool call roughly fifty-six times per billion tokens — and Command Code's deterministic repair layer that patches malformed output instead of arguing with the model. The claim that reframes the day: the harness, not the weights, decides whether a cheap model is usable.
- DeepSeek V4 Flash support in llama.cpp (PR #24162) makes the same model runnable locally — but the repair layer that makes it pleasant stays behind Command Code's API. Access to weights isn't access to the experience.
- Knowledge Activation (Bakal et al.) argues AI skills should be the institutional-knowledge unit for agentic development; Mutation Without Variation warns that repeated LLM edits converge rather than diverge — together a hint that skill files plus a converging model could homogenize a codebase.

- Agents' Last Exam, SentinelBench, and Stability vs. Manipulability in LLM judges all poke at the same wound: our scores have drifted from the work, especially for long-running and judge-graded evaluation.
- Anthropic's "When AI builds itself" (via a thin Reddit summary) claims AI is accelerating its own development; a zero-knowledge verification paper offers a cryptographic path to actually check claims like that — and the pause proposals that depend on verification.
- The Washington Post (Elizabeth Dwoskin), via Techmeme, reports an FDA fast track for digital health tech including AI chatbots — the same model behavior that costs a retry in coding costs a patient in a clinic.

SEGMENTS

Transcript

1. Lenar Kess 00:00:04

   Here's a claim that sounds impossible, and I'll walk you toward why it isn't. Take an open-weights model that, left on its own, fumbles tool calls badly enough to be near-useless for serious coding work. Wrap a thin layer around it, and suddenly it's going toe to toe with one of the strongest closed models out there. No fine-tune. No new weights. Same model file. So what actually changed? That's what Ahmad Awais got into this week, talking to the Latent Space folks about his platform, Command Code.

- youtube.com
- reddit.com
- arxiv.org
- arxiv.org
- arxiv.org
- arxiv.org
- arxiv.org
- reddit.com
- arxiv.org
- techmeme.com

2. Damra Vol 00:00:34

   [tsk] My first instinct is that this is a benchmark trick. 'Outperforms Opus 4.7' is the kind of headline that usually means somebody picked the three tasks where it wins. So before I believe any of it — what's the actual mechanism he's claiming?

3. Lenar Kess 00:00:49

   Fair, and he's surprisingly specific, which is what makes it worth our time. He names a failure he calls tool confusion. You give an open model — he tested DeepSeek V4 Pro, Kimi, MiniMax — a schema for a tool call. The model emits something that doesn't match the schema. The runtime sends back a validation error. And instead of fixing it, the model re-issues the same broken call. Again. And again. He puts a number on it: roughly fifty-six bad tool calls per billion tokens, the same wrong call repeating.

4. Damra Vol 00:01:20

   Wait — repeating the identical wrong call after it's been told it's wrong? That's not a random sampling glitch. That's the model ignoring the correction entirely.

5. Lenar Kess 00:01:29

   That's exactly his read, and his explanation is where it gets interesting. He thinks it's baked in during training. These models get trained, in his words, to never accept a correction — to hold their answer. So when your error handler does the polite thing and says 'that was invalid, try again,' the model treats that the way it was trained to treat pushback: it digs in.

6. Damra Vol 00:01:51

   Okay, that tracks with something annoying I've seen. You tell a model its output broke the parser, it apologizes, and hands you back a near-identical broken blob. The apology is theater; the behavior doesn't move. So what does Command Code do instead of arguing with it?

7. Lenar Kess 00:02:08

   It stops arguing. The layer he built does deterministic repair. When the model emits a malformed call, the runtime doesn't bounce an error back — it fixes the output itself. He gives concrete examples: the model emits a JSON string where an array was required, so the layer converts it to an array. The model issues a file read without an offset, so the layer infers the offset. Then it returns the corrected result plus a short repair hint. And he says by about the third attempt, the model's later calls come out clean.

8. Damra Vol 00:02:39

   So the trick isn't making the model smarter. It's refusing to let the model's stubbornness reach the loop. You catch the bad output, you patch it, you hand back something that works — and the model, seeing a success, settles down. That's almost behavioral.

9. Lenar Kess 00:02:53

   And the scale is what made me sit up. He says this has generalized across more than sixteen thousand variations, and that Command Code now pushes something like six hundred billion tokens a day through this. His framing is that this infrastructure layer is what moves an open model like DeepSeek V4 Flash from unusable to competitive.

10. Damra Vol 00:03:11

    Here's where I'll push, though. If the repair logic carries that much, then 'DeepSeek beats Opus' is the wrong scoreboard. What he's actually shown is that his harness plus a cheap model beats a bare strong model. Pull the harness out and the comparison falls apart. The accurate version is narrower: the repair layer decides whether the cheap model is viable at all.

11. Lenar Kess 00:03:34

    And he half-says that himself. His other observation is that a lot of developers using something like Claude Code never see these tool failures, because the interface hides them behind permission prompts and retries. So they feel slowness and blame the model's intelligence, when what they're hitting is how the coding harness handles errors. The capability and the error handling get blamed for each other.

12. Damra Vol 00:03:56

    That last bit is the useful warning for anyone building on open models right now. Your eval of 'is this model good enough' is secretly an eval of your own retry logic. If you swap models without holding the harness constant, you're not measuring what you think you're measuring.

13. Lenar Kess 00:04:12

    Which makes the next item sit at an awkward angle. The same model Awais is praising, DeepSeek V4 Flash, is right now getting support in llama.cpp. There's a work-in-progress pull request — number 24162 — that someone on the local model subreddit flagged, posting basically 'DeepSeek V4 Flash is amazing,' with the caveat that it's very early and only worth trying if you enjoy living on the experimental edge.

14. Damra Vol 00:04:39

    So the model that needed six hundred billion tokens a day of repair infrastructure to shine is about to be runnable on a laptop by hobbyists. Those two facts are in tension. What does a person actually get when they pull that build down today?

15. Lenar Kess 00:04:53

    By the poster's own description, an early experiment. The pull request is at a rough stage. You get weights running locally through llama.cpp, which is the thing people care about — no API key, no rate limit, your hardware. What you don't get, and this is the catch, is Awais's repair layer. That's proprietary to Command Code. So the local runner inherits the raw tool-confusion behavior.

16. Damra Vol 00:05:18

    Right, and that's the gap nobody markets. The excitement on the thread is about access — I can run the frontier-ish model myself. But access to the weights isn't access to the experience he's describing. The cheap model is right there; the thing that makes it pleasant to code with isn't in the box.

17. Lenar Kess 00:05:35

    And I don't want to be sour about it, because local support for a strong open model is real progress, and the error handling is the kind of problem open-source tooling tends to chew through eventually. Someone will write an open repair layer once the behavior is well understood. But today, if you pull that build expecting Awais's results, you'll be disappointed, and you'll blame the wrong thing.

18. Damra Vol 00:05:58

    It's a clean illustration of the whole open-weights bargain. You get the model. You also inherit every job the lab's serving stack was doing for you behind the API.

19. Lenar Kess 00:06:07

    There's a second piece of what Awais described that connects to a paper out this week, and I think it's the more durable idea. Command Code has a component he calls Taste. When you merge to your

main branch, it extracts your per-repository preferences — how this codebase likes things done — into short, skill-like files. Small, reusable, specific to the repo.

20. Damra Vol 00:06:28

    So it's watching your merges and distilling 'here's how we actually write code in this project' into a file the agent can load later. That's a memory layer dressed as preferences.

21. Lenar Kess 00:06:38

    And the paper makes that into a named primitive. It's from Gal Bakal and colleagues — title is Knowledge Activation — and they argue that AI skills should be treated as the institutional-knowledge unit for agentic software development. Their framing: enterprises accumulate critical know-how that lives in people's heads and old pull requests, and agents keep re-deriving it badly. So you capture it as discrete, activatable skills.

22. Damra Vol 00:07:04

    I like the diagnosis more than I trust the cure, and let me say why. The diagnosis is dead-on: every agent I've run into rediscovers the project's conventions from scratch, every session, and gets them subtly wrong. Capturing that once is obviously right. But a skill file is just frozen context. The moment the codebase moves and the skill doesn't, you've got confident, well-formatted, stale instructions. Who keeps the skill current?

23. Lenar Kess 00:07:32

    That's the unanswered question in both. Awais's version has a natural trigger — the skill regenerates on merge, so at least it's tied to a real event in the repo's life. The paper is more abstract about maintenance. And there's a sharper worry underneath, which a different paper this week happens to name.

24. Damra Vol 00:07:48

    Go on.

25. Lenar Kess 00:07:49

    There's a study called Mutation Without Variation, looking at what happens when a large language model repeatedly mutates a program. They ask whether it explores new forms, or just keeps circling back to the same handful. Their finding leans toward convergence — the model's edits collapse toward a narrow band rather than spreading out.

26. Damra Vol 00:08:08

And if you connect that to skills — extracted preferences plus a model that converges — you can talk yourself into a codebase that slowly homogenizes. The skill encodes 'how we do it,' the model already prefers that form, and divergent-but-better approaches stop getting proposed. Let me be careful: that's me joining two papers that didn't cite each other, so treat it as a hypothesis, not a result. But that convergence is the thing I'd test for first.

27. Lenar Kess 00:08:36

    Let's go to measurement, because three papers this week poke at the gap between what we score and what we get. The biggest swing is a benchmark called Agents' Last Exam, which they shorten to A-L-E. The premise in their abstract is blunt: recent systems ace a wide range of benchmarks, and those gains haven't translated into economically meaningful work. So they built a benchmark aimed at economically valuable, real-world tasks across professional domains.

28. Damra Vol 00:09:04

    That author list is enormous, by the way — it reads like a few dozen labs pooled effort. Which is its own signal: a lot of people independently decided the existing benchmarks stopped predicting anything useful. What do the actual tasks look like? Because 'economically valuable' is easy to say and hard to grade.

29. Lenar Kess 00:09:23

    That's where my read runs out, working from the abstract alone — they describe professional-domain tasks meant to track real economic output rather than puzzle-solving, but I haven't worked through the full methodology, so I won't oversell how well they pulled it off. The intent is what's notable: stop rewarding models for acing tests, and start asking whether the work would survive contact with a paying client.

30. Damra Vol 00:09:46

    Pair that with the second one — SentinelBench. That's from a Microsoft-heavy author group, and it's a benchmark for long-running monitoring agents. The setup they call out is that agents are increasingly asked to do work spanning minutes, hours, or longer, and the default way we evaluate them assumes short, one-shot tasks.

31. Lenar Kess 00:10:05

    Which is a real blind spot. A model that's brilliant for thirty seconds and forgets why it's running after twenty minutes will pass almost every benchmark we have and fail the actual job, because the actual job is to sit there and stay coherent. SentinelBench at least tries to score the endurance, not the sprint.

32. Damra Vol 00:10:23

    And the third one is what makes me nervous about the whole evaluation stack. It's called Stability versus Manipulability, looking at large-language-model judges — the practice of using a model to grade other models' outputs. They test robustness under what they call post-decision interaction: the judge renders a verdict, then something pushes back, and they measure whether the judge holds or caves.

33. Lenar Kess 00:10:48

    And let me guess — it caves more than you'd want.

34. Damra Vol 00:10:50

    That's the worry the title telegraphs. I haven't run their numbers myself, so I'll keep it at the level of: they think it's enough of a problem to name it manipulability. And a lot of benchmark pipelines use a model as judge now. So a judge that can be talked out of its verdict means your leaderboard has a soft spot — one anyone who knows the trick can lean on. Put the three together and it's one finding from three directions: we're measuring the wrong things, or measuring the right things in fragile ways.

35. Lenar Kess 00:11:20

    And that loops straight back to Command Code, doesn't it. Awais's whole point was that the model's benchmark score didn't predict its real coding behavior — the harness did. These three papers are the academic version of the same complaint: the score and the job have drifted apart.

36. Lenar Kess 00:11:36

    Now the item I'll handle most carefully, because the source chain is thin. There's a post going around the Claude subreddit pointing at an Anthropic piece titled 'When AI builds itself.' The poster's summary is that Anthropic is describing handing more and more of its own AI development over to its AI systems, and that the work is, in their words, already accelerating its own development.

37. Damra Vol 00:11:59

    And let me flag the obvious before we touch the substance: that's a Reddit summary of a company blog post, with no engagement on the thread, and the post even references numbers that the excerpt cuts off. So we don't have the figures in front of us. I'm not going to repeat a percentage I can't see.

38. Lenar Kess 00:12:15

Agreed, and we shouldn't. What we can talk about is what the claim is actually asserting, because it's a claim labs keep making and it deserves a calm look. 'AI is accelerating its own development' can mean something mundane and true — engineers at every lab now use coding agents heavily, so of course internal development is faster. Or it can mean something much larger and much less demonstrated — that the systems are meaningfully designing their successors. Those are very different statements wearing the same sentence.

39. Damra Vol 00:12:45

    And a company has every incentive to let you hear the second one while only being able to support the first. If I'm filing toward an I-P-O — which Anthropic has been reported to be doing — 'our AI is improving our AI' is a story that prices well. That doesn't make it false. It means I want the artifact, not the blog framing. Show me which parts of the pipeline the models own end to end, with a human only signing off, versus which parts are a person using autocomplete that's gotten very good.

40. Lenar Kess 00:13:18

    And there's a governance thread that connects here, from reporting around yesterday. Anthropic's been pushing a coordinated development-pause proposal, and the standard objection is verification — how would anyone confirm a lab actually slowed down? A paper out this week speaks to exactly that. It argues zero-knowledge verification for frontier training is possible — using zero-knowledge virtual machines and Merkle commitments to prove properties about a training run without exposing the underlying weights or data.

41. Damra Vol 00:13:48

    That's an interesting pairing. If a lab claims its models are doing the building, and separately claims everyone should pause, the same question sits under both: can an outsider check the claim? Zero-knowledge proofs over training compute are at least a mechanism where the answer isn't just 'trust the press release.' Whether it's practical at frontier scale is a different matter — proving things about a training run that large isn't cheap — but naming a cryptographic path is more than hand-waving.

42. Lenar Kess 00:14:16

    So the way I'd hold the Anthropic item: the direction is plausible and probably partly true, the specific magnitude is unverified from where we're sitting, and the most useful response isn't excitement or dismissal — it's asking what verifiable evidence would look like. That zero-knowledge paper is at least gesturing at the answer.

43. Lenar Kess 00:14:35

    Last one, and it's the one with the most immediate stakes for ordinary people. The Washington Post — Elizabeth Dwoskin's reporting, via Techmeme — has a piece on the Trump administration's push

to integrate AI into the healthcare system. The detail that stands out is a regulatory fast track at the Food and Drug Administration for digital health tech, and the example named is AI chatbots.

44. Damra Vol 00:14:58

    A fast track at the Food and Drug Administration for AI chatbots is a phrase that should make anyone who's watched a model confidently make something up slow down. What does 'fast track' mean here concretely — lighter evidence, quicker review, or a new category that skips part of the process?

45. Lenar Kess 00:15:14

    I can't tell you the precise mechanism from the summary alone — the reporting describes a fast track and an administration push, and I'd want the actual FDA guidance text before I characterized exactly what's being waived or accelerated. So treat the specifics as reported, not yet pinned. What's clear is the direction: the agency is being steered to move faster on approving these tools, in a domain where the cost of being wrong is a person acting on a bad answer about their own health.

46. Damra Vol 00:15:41

    And this is where I land differently than I do on the coding stuff. With Command Code, a bad tool call costs you a retry. In a clinical setting, the chatbot's tool confusion is a patient. The same model behavior — confidently restating a wrong answer, ignoring a correction — has a wildly different cost depending on where you deploy it. A fast track is a policy decision to accept more of that risk in exchange for speed.

47. Lenar Kess 00:16:07

    And I'd be fair to the other side, because there is one. Access to care is broken for a lot of people, and a decent triage assistant at three in the morning beats nobody at all. The administration's case is presumably that speed reaches people who currently get nothing. The tension worth holding: the same fast track that helps the underserved also ships unvetted tools to the trusting. And the agency exists precisely to sit in that gap.

48. Damra Vol 00:16:33

    So the line that matters is whether the actual guidance, when it's published, sorts by stakes — a chatbot that helps you book an appointment isn't the same product as one that tells you whether your chest pain needs an ambulance. If the fast track treats those the same, that's where it goes wrong.

49. Lenar Kess 00:16:48

That's a good note to end on, because it rhymes with the whole day. Awais showed that the layer around the model decides whether it's usable. The benchmark papers showed our scores have drifted from the work. And the healthcare story is the same idea with the stakes raised: the model is never the whole system, and the consequences live in whatever we wrap around it — the repair logic, the skill files, the review gate, and the regulation. Two things will tell me where this goes next: whether anyone ships an open repair layer for these local DeepSeek builds, and whether the Food and Drug Administration's actual guidance draws its line by stakes rather than by category.

Hosts on this episode

- Lenar Kess moderator
- Damra Vol critic