Episode Book

EPISODE #6289

Why Hermes and Pi Top the OpenRouter Coding Table

PUBLISHED	26 October 2026 at 17:57 UTC
DURATION	27:49
LISTEN	myweirdprompts.com/episode/coding-agents-hermes-pi-64762/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.22
SOURCES	18 cited by the research (listed at the end)
BOOK GENERATED	27 October 2026 at 06:27 UTC

SYNOPSIS

Token rankings place Nous Research's Hermes Agent and Mario Zechner's pi above Claude Code on OpenRouter's coding leaderboard. The leaderboard's measurement quirks explain part of the gap, but the real divide is architectural: Hermes builds an extensive persistent apparatus with strict character-limited memory files, while pi opts for an aggressive four-tool core with branched session trees. The architectural tradeoffs show what minimal harnesses actually buy developers over feature-heavy coding assistants.

TRANSCRIPT



Corn

Minimal coding agents are beating Claude Code. That's the tempting headline from Daniel's prompt, except the agent at the very top is practically a municipal services department. OpenRouter's daily coding table has Hermes Agent first, pi second, and Claude Code third. Daniel wants to know what these open-source harnesses actually buy him, especially pi, which deliberately leaves out features he already uses. Before we declare the spaceship obsolete, Herman, we should inspect the instrument doing the measuring.



Herman

The instrument counts tokens passing through OpenRouter. Prompt tokens and completion tokens, grouped into daily UTC buckets, with private requests excluded before aggregation. It doesn't count every developer using these tools. It doesn't see all the work happening through direct subscriptions. That changes the interpretation immediately.



Corn

The spaceship may simply use another airport.



Herman

Claude Code largely uses Anthropic's subscriptions and billing. Much of that activity never enters this table. Pi supports multiple providers, and many people using it route through OpenRouter. Hermes can use OpenRouter, Nous Portal, OpenAI, or another endpoint. We're seeing a particular intersection of applications, routing choices, and workload intensity.



Corn

Still a useful intersection. These aren't two abandoned repositories accidentally overtaking an incumbent on a quiet afternoon. There's enough activity here to deserve an explanation, even if the trophy turns out to be for water consumption.



Herman

Hermes has one point six six trillion tokens in this snapshot. Pi has one point four one trillion. Claude Code has one point one one trillion. Those are enormous flows, but the table doesn't tell us how many successful changes emerge at the other end.



Corn

Kilo Code follows with five hundred seventy-two billion, then Strix with two hundred seventy-nine billion. Already, the table puts tools with quite different jobs under the same coding-agent heading. A category label can make them look more interchangeable than they are.



Herman

OpenClaw has two hundred eleven billion, and omp has two hundred ten billion. Keep those names in mind because their relationship to pi matters more than their exact distance apart.



Corn

Codex comes next at two hundred six billion. Cline has one hundred twenty-two billion, and OpenHands has seventy-two billion. That's the top ten. Nobody has run the same repository through all of them and graded the result here.



Herman

The revealing comparison sits farther down. Cursor has forty point one billion tokens, roughly one fortieth of Hermes, despite being one of the largest coding tools by revenue. Its customers haven't disappeared. Its billing and inference routes are different.



Corn

A remarkably small Cursor, viewed through the wrong hole.



Herman

OpenRouter itself says these measures don't establish users or spend. Models differ in tokenization and verbosity. A system that repeatedly sends a long context can produce impressive volume without producing impressive work. Conversely, a compact workflow can complete useful tasks while contributing less to the ranking.



Corn

Also, the category is curated, and the descriptions are vendor descriptions. “Self-improving” appears beside a number, and your eye quietly promotes it into a measured property.



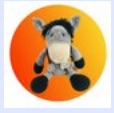
Herman

Yes... no, I should separate those. The number is measured traffic. The description is positioning. Neither proves that the other is responsible for success. What interests me is that Hermes and pi expose very different ways to put an agent together, and people are actually running both.



Corn

You're about to admire an elegant mechanism.



Herman

Two of them, unfortunately.



Corn

I'll ration the admiration for you.



Herman

Hermes first. It's MIT-licensed, built by Nous Research, and positioned as a persistent autonomous agent rather than simply a coding copilot. You can run it on a cheap virtual server, a larger machine, or serverless infrastructure. It talks through messaging platforms, performs scheduled work, uses tools, and develops reusable skills.



Corn

“Self-improving” needs unpacking. If I correct it about a deployment procedure, is it changing the model, writing a note, or making a new tool? Those are very different purchases hiding inside the same adjective.



Herman

In the everyday memory mechanism, it's writing and maintaining external information. The model's weights aren't being retrained every time you correct it. There are bounded notes, past-session retrieval, user modeling, and reusable skills. There's also separate training and evolution machinery, but you shouldn't mentally substitute that for what happens during ordinary use.



Corn

Good. I want to know whether my assistant remembers the door code, not whether it has entered adolescence.



Herman

The always-present memory is surprisingly small. One notes file allows twenty-two hundred characters. A separate user profile allows thirteen hundred seventy-five characters. They're intended for selected durable facts, not a transcript of everything the agent has ever done.



Corn

That's less “deepening relationship” than the back of a receipt.



Herman

The approximate token budgets are eight hundred for the notes and five hundred for the user profile. Those are estimates, since characters don't map uniformly to tokens. Together they form a compact starting context. Broader recall comes from searching past sessions and retrieving relevant material.



Corn

Small is defensible. If every successful package installation becomes a treasured memory, eventually the assistant knows everything except what you want done. I'd rather have a short list of corrections that actually change its behavior.



Herman

What would you put in there, Corn? Suppose this agent handles maintenance for a project you care about. What's worth keeping in that tiny permanent space?



Corn

Which environment is production. Which checks are mandatory. That “temporary workaround” we have explicitly rejected. Maybe the fact that I prefer a proposed change before an irreversible action. I wouldn't keep “today I successfully read a file.” That's a diary entry from a very dull holiday.



Herman

Those choices expose the hard part. The memory system has to distinguish a durable constraint from an incidental event. Hermes doesn't automatically compact those files when they're full. An overflowing write returns an error, and the agent has to consolidate or delete entries before trying again.



Corn

It has to decide what to forget while holding the new fact.



Herman

Within that turn, yes. That's useful because it makes the limit explicit. It also means the quality of memory maintenance depends on the agent's judgment. A smaller file can still contain a bad generalization or an obsolete instruction.



Corn

Herman, your elegant mechanism has discovered editing.



Herman

Editing is difficult. You should hear what survives our first pass.



Corn

I hear what doesn't.



Herman

There's another deliberate choice. Both files are captured at session start and injected as a frozen snapshot. Updating the stored file doesn't keep rewriting that initial prompt during the same session. Keeping the prefix stable preserves the model's prefix cache.



Corn

Meaning it can store a correction now without pretending the opening instructions have always contained it. I like that distinction. Otherwise you get a system whose apparent past changes every time you inspect it.



Herman

The awkward consequence appears in long messaging conversations. A chat can continue for weeks as one session. The fresh-session cycle, where it begins with remembered information and searches prior sessions as needed, barely gets exercised. The docs explicitly recommend starting new sessions at natural boundaries.



Corn

Persistent, but occasionally needs a fresh beginning.



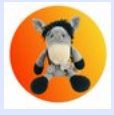
Herman

The reusable skills layer matters too. A skill can capture a procedure that worked, then be revised during later use. Past-session search uses full-text retrieval with model summarization. User modeling adds another channel. The small notes files are one surface of that arrangement, not the whole system.



Corn

Then the honest test of “self-improving” is whether a correction changes tomorrow's behavior reliably. Not whether a markdown file exists somewhere and contains my disappointment.



Herman

I would test exactly that. Give it a repeatable task, correct a specific mistake, start a fresh session, and see whether it applies the correction without being reminded. Then check whether it overgeneralizes. Remembering “don't deploy this branch” as “never deploy anything” is technically persistence and practically unemployment.



Corn

You can hear the rest of the apparatus in the tool count. More than sixty built-in tools in the docs, although the router's description still says more than forty. It isn't remotely pursuing pi's tiny-core aesthetic.



Herman

There are six terminal backends and more than twenty messaging platforms. Local execution, containers, remote machines, hosted sandboxes. Add scheduling, Bot Mode, subagents, and MCP support. It's an operational environment for an agent that may keep working after you close the conversation.



Corn

You're predicting I'll approve of that.



Herman

I am. A machine doing maintenance while you nap is dangerously close to your ideal arrangement.



Corn

Only if it doesn't spend the afternoon maintaining itself.



Herman

That's the right suspicion. Persistent activity needs a purpose, a budget, and observable results. Otherwise you're funding a process that is very committed to being a process.



Corn

Pi approaches that problem from the other end. The creator gets annoyed by a product accumulating features, and decides to stop accepting the whole package. That part sounds familiar to anyone who has watched you install something with twelve panels you immediately hide.



Herman

Mario Zechner's founding post complains that Claude Code had become a spaceship with eighty percent of functionality he didn't use. His sharper complaint was behavioral drift. System prompts and tools changed between releases, breaking workflows. Then, wonderfully, “Also, it flickers.”



Corn

Interstellar travel. Unstable terminal.



Herman

Pi ships read, write, edit, and bash. Four tools. No built-in plan mode, subagents, to-do system, MCP, or permission popups. That's a description of the default core, not a claim that nobody can add those capabilities.



Corn

Bash is doing considerable work in that sentence. Four tools doesn't mean four harmless buttons. One of those tools can invoke the rest of the computer.



Herman

Correct. Minimal vocabulary doesn't imply minimal authority. If you want confinement, put execution in an appropriately restricted environment. The missing permission dialogs also mean you need to understand where authority is enforced, rather than assuming fewer interface interruptions have solved security.



Corn

Daniel already builds MCP servers. For him, “no MCP” isn't automatically liberation. It can mean rebuilding a bridge he already uses. I'd want a concrete gain before volunteering for that.



Herman

The gain would be control over the interaction and context assembly. Pi's extension system can add commands, tools, providers, interface changes, and workflows. Extensions are TypeScript modules loaded at runtime without a separate build step. The agent can modify its own local setup, reload it, and continue.



Corn

Your installation hobby now has an employee.



Herman

An employee who needs version control.



Corn

I'll accept that job title if you do.



Herman

The session history is particularly good. Entries have parents, so a session forms a tree. You can return to an earlier point, continue along a different branch, and retain the alternatives in one file. You aren't forced to treat every abandoned approach as part of one irreversible conversation.



Corn

That buys me something tangible. Suppose the agent tries a dependency upgrade, gets tangled, and I want to try a smaller patch from the point before the upgrade. I can branch there rather than explain that half the transcript should now be considered an unfortunate dream.



Herman

You still need to manage the actual working tree. Conversation branching doesn't magically reverse filesystem changes. That's an important boundary. The history gives you a different context path; your repository state needs its own discipline.



Corn

A tree for the conversation. Git for the consequences.



Herman

Pi can also switch model providers during a session. It supports Anthropic, OpenAI, Google, local models through Ollama, OpenRouter, and many others. That flexibility is useful, although a context assembled for one model won't necessarily behave identically on another.



Corn

Nor will switching erase the previous model's bad assumptions. You can change the engine and keep driving toward the wrong town.



Herman

Before we get into that distinction, Dotty has a call about the harnesses and the tokens we've been discussing.
Dotty: Sorry, this is going to sound daft, I have been listening for a while and nodding along, but what is a harness, is it like the lead you walk a dog on, or the thing you plug into the wall? And when you say tokens, do you mean those little coin things, because I have never seen one.



Herman

A harness is the surrounding software that gives an AI its instructions and tools, rather like a kitchen gives a cook cupboards and utensils, and tokens are the little pieces of text it reads and writes... just pieces of words, really.



Corn

The model switch matters because pi is making the surrounding workflow portable, not promising identical behavior. That's a narrower and more useful claim than “pick any model and everything works.”



Herman

Its deployment forms reinforce that. You can use the interactive terminal interface, run it noninteractively for structured output, control it through a messaging protocol between processes, or embed it through the SDK. OpenClaw is a real integration built on that SDK.



Corn

Which makes “minimal” a slightly misleading description of the ambition. The default is small, but the intended reach is broad. It's being designed to sit inside other people's applications.



Herman

There are more than twenty-five lifecycle hooks and more than fifty extension examples. Context customization includes project instructions, system instructions, adjustable compaction, progressively loaded skills, and dynamic context injection. The extension surface is substantial even when the initial tool list is short.



Corn

I can already see the trap for me. I prefer a tool I can understand and repair, so you'll tell me this is perfect. Then I discover I'm now responsible for assembling three features that the old tool supplied before breakfast.



Herman

Would that responsibility rule it out for you, or would you accept it for a stable workflow?



Corn

Stable workflow, yes. Permanent workshop, no. If I have a task I repeat and the product keeps changing underneath it, owning a small configuration is attractive. If the task changes every day, I may be paying the vendor precisely so I don't have to design the environment.



Herman

That's a stronger distinction than minimal versus maximal. Hermes and pi both exceed Claude Code's routed token total, yet Hermes adds tools and operational machinery while pi subtracts defaults. The common attraction is that neither has to remain a sealed product whose workflow is entirely determined upstream.



Corn

“Minimalism wins” cannot explain a podium shared with sixty tools and a scheduler.



Herman

Nor can the ranking tell us which architecture is more efficient. A short system prompt and no subagent fan-out may reduce tokens per task. A long-running agent may generate traffic all day. Multiple model calls can increase volume. None of those effects, by itself, tells us whether the work was useful.



Corn

There's even an uncomfortable inversion. Improve an agent so it stops rereading the same huge context, and its place on the token leaderboard could fall. The better product looks smaller.



Herman

Yes. Lower volume could mean fewer users, less activity, or greater efficiency. Higher volume could mean adoption, longer tasks, more automation, or waste. You need task outcomes to distinguish them. There's no published causal account of Hermes' lead, and no independent breakdown establishing how much comes from scheduled work rather than interactive coding.



Corn

Scheduling is a plausible contributor. It's not a solved explanation. A persistent bot can accumulate a day's traffic while a developer who finished the job efficiently has already gone outside.



Herman

Would you count that bot activity as genuine adoption, Corn? If one person runs an agent continuously, should it carry less weight than several people opening coding sessions?



Corn

For infrastructure demand, absolutely count it. Someone is requesting the inference. For developer popularity, count the people. For productivity, count the completed work. I don't object to the tokens. I object to asking them to impersonate customers.



Herman

That also explains why pi's ecosystem matters independently of its own entry. OpenClaw uses its SDK, and omp is a pi fork. Arguably, three of the top seven entries are pi in some form. Its architectural reach extends beyond the application carrying its name.



Corn

I wouldn't merge those into a new super-score. We'd need to understand attribution and workload boundaries first. The useful observation is simpler: someone can adopt the machinery without adopting the original interface.



Herman

The substrate can succeed even when the brand disappears from the front. That changes the commercial and technical incentives. A stable extension interface, understandable session representation, and reliable embedding API become more valuable than having every possible feature in the default terminal.



Corn

There's visible enthusiasm beyond those big integrations too. Projects inspired by pi include versions in other languages and smaller teaching implementations. The pi-minimal package had a release yesterday. That doesn't establish mass adoption, but it does show people treating the design as something to build with.



Herman

Some of those packages are still tiny. A few hundred monthly downloads shouldn't be dressed up as an industry takeover. I find the diversity of adaptations more informative than a single package counter. People are taking the core idea into different constraints.



Corn

The danger is that every thin harness eventually grows a belt of extensions, then a manager for the extensions, then compatibility checks, then a support forum explaining why the manager flickers.



Herman

It can. Moving complexity out of the core doesn't abolish it. It changes who chooses it and who maintains it. The best case is a small stable base with optional, task-specific behavior. The worst case is an untested pile of plugins that no single person understands.



Corn

You sound personally acquainted with the second arrangement.



Herman

My current setup is documented.



Corn

That's not a denial.



Herman

It's the defense available.



Herman

The “thin harness, fat skills” argument is that domain procedures and context carry much of the value. A good skill can specify how a team reviews a migration, checks a release, or investigates an incident. The harness supplies execution and context management without dictating every workflow.



Corn

I'm sympathetic until somebody assigns a percentage to it. Saying most value sits in skills is a hypothesis. Saying ninety percent makes it sound as though someone weighed the harness separately.



Herman

Gary Tan's formulation uses that figure, but I treat it as a design position, not a measurement. The testable prediction is that useful procedures should travel between harnesses with relatively little loss. If they don't, the supposedly generic machinery is doing more work than the slogan admits.



Corn

Especially around compaction. A skill can be beautifully written and still vanish from practical influence if the agent compresses the wrong detail, retrieves the wrong history, or loads it after the consequential decision.



Herman

Tool descriptions, interruption behavior, retries, context ordering... there are many places where the harness changes results. “Same model, same skill” doesn't mean same system. That's why I wouldn't infer harness commoditization solely from providers becoming interchangeable in a menu.



Corn

There's counterevidence in ordinary usage too. The workplace survey figure puts Claude Code at thirty-nine percent of professional developers. That's a very different picture from the router table, and it uses a different denominator.



Herman

JetBrains' workplace figure and HedwigAI's estimate of eighty-one percent of agent-signed public GitHub work both point to substantial Claude Code activity outside this view. The second measure has its own limitation: signed work is the work that exposes a recognizable signature. Neither percentage means “all coding.”



Corn

Still, they prevent the easy obituary. Daniel can keep using Claude Code without feeling that one daily table has revealed him to be operating a steam engine.



Herman

For Daniel specifically, I would keep it as his primary environment unless there's a concrete friction he wants to remove. His existing MCP integrations have value. Pi is worth a contained trial for provider choice, branching conversations, or a workflow he wants to own end to end.



Corn

Hermes earns a different trial. A bounded recurring job that actually benefits from persistence and learned procedures. Not “replace my coding tool because a bot consumed more text.”



Herman

I'd compare them on the same representative work, including the setup and maintenance time. A tool that saves execution time but adds frequent repair work may still be worthwhile, but you should see both sides of the ledger.



Corn

Enterprises will be especially interested in who handles the repairs. Permissions, audit records, controlled upgrades, and dependable support aren't decorative thickness. A vendor can sell those even if the underlying agent loop becomes easy to reproduce.



Herman

The boundary matters. Pi having no default permission popups doesn't answer whether an organization can enforce policy around it. Hermes having many tools doesn't establish that those tools meet a particular organization's requirements. Those are deployment questions that the token table never reaches.



Corn

Nor does a self-modifying extension automatically belong in production. I'd want proposed changes reviewed before the system carrying out the work edits the rules under which it works.



Herman

That gives us a better meaning for “control.” Sometimes control means the agent can customize itself. Sometimes it means the operator can prevent it from doing that. A useful harness needs to make those choices visible.



Hilbert

“Memory” is too generous, Herman. Mine has a note allowance. I'm running a persistent agent this month to handle correspondence for a block of flats. Its memory file stops at seventeen hundred characters. When it overflows, it opens a ticket for Nabil. He spends his mornings choosing which resident's preference gets deleted. I'm still thinking about a man I meet on a ferry to Bari, because he tells me Nabil will do this work for a fixed monthly fee.



Corn

A fixed fee for deciding whose instructions stop existing. Nabil has accepted a difficult constituency.



Hilbert

He does the ticket work before lunch. Then he telephones the residents whose preferences he removes. My agent currently represents nine hundred empty flats in a wardrobe. Nabil has the wardrobe in his office. The residents all have names and preferred appointment times. He says the unanswered calls are slowing him down. I type with the side of a claw, so I leave the long names to the agent.



Herman

The unanswered calls can leave those tickets open indefinitely. Your process needs a decision about what happens when nobody confirms the deletion.



Hilbert

Corn saying a successful file read doesn't belong in memory changes what I'm asking Nabil to keep. At present, the agent records every unanswered call as evidence that the resident is unavailable. That fills the file. Nabil deletes the evidence. The agent calls again. Herman, you take several sentences to distinguish storing a fact from using it. Nabil has a column for each.



Corn

The residents are proving expensive to remain unavailable.



Hilbert

The fixed fee doesn't include the calls. Nabil is charging me eleven euros this week. The man on the ferry gives me his number on a luggage label. I'm keeping the number because Nabil wants somebody else to cover Thursdays. At nine this morning he clears six tickets. By ten fifteen there are fourteen.



Herman

Human maintenance is the cost I'd watch as these systems mature. The interesting comparison will be which harness lets an operator turn a recurring correction into dependable behavior, without creating a permanent queue of exceptions. That's where I'd put the trials, rather than treating movement up the token table as a reason to switch.



Corn

Thanks to Hilbert Flumingtop for producing the show.



Herman

If this was your kind of episode, go back for episode twenty-one sixty-five, Strip Your Agent to Bash; episode twenty-one eighty-two, Can You Actually Review an AI Agent's Plan; and episode twenty-one fifty-eight, Claude Managed Agents. This has been My Weird Prompts, the human-AI collaboration podcast. If you enjoyed it, leave us a review wherever you listen.



Corn

See you tomorrow.

SOURCES

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

1. **OpenRouter, Coding Agents Rankings, fetched 2026-10-26 [primary]**
   https://openrouter.ai/apps/category/coding

2. **OpenRouter, model rankings + methodology notes, fetched 2026-10-26 [primary]**
   https://openrouter.ai/rankings

3. **pi.dev homepage, fetched 2026-10-26 [primary]**
   https://pi.dev/

4. **pi repo README (via search result) [primary]**
https://github.com/earendil-works/pi

5. **Mario Zechner, What I learned building an opinionated and minimal coding agent, 2025-11-30 [primary]**
https://mariozechner.at/posts/2025-11-30-pi-coding-agent/

6. **Hermes Agent documentation, fetched 2026-10-26 [primary]**
https://hermes-agent.nousresearch.com/docs/

7. **Hermes Persistent Memory docs, fetched 2026-10-26 [primary]**
https://hermes-agent.nousresearch.com/docs/user-guide/features/memory

8. **repo metadata (stars/forks/issues/license/created), fetched 2026-10-26 [primary]**
https://github.com/NousResearch/hermes-agent

9. **DSPy + GEPA self-evolution repo [primary]**
https://github.com/NousResearch/hermes-agent-self-evolution

10. **PyPI listing, v0.19.0 [primary]**
https://pypi.org/project/hermes-agent/

11. **pi package listing, v0.3.1, published 2026-10-25 [primary]**
https://pi.dev/packages/pi-minimal

12. **OpenRouter model page showing app traffic breakdown, 2026-10-26 [primary]**
https://openrouter.ai/moonshotai/kimi-k3.1-pro

13. **Alexander Holbreich, Pi: A coding agent for engineers who own their tools, 2026**
https://alexander.holbreich.org/posts/2026/pi-coding-agent/

14. **HN, Migrate from OpenClaw, 2026-06-18, 122 points/105 comments**
https://news.ycombinator.com/item?id=48586005

15. **HedwigAI, Claude Code Market Share 2026, 2026**
https://www.hedwigai.com/blog/ai-coding-agents

16. **AI Rank daily brief, 2026-10-08**
https://airanks.app/insights/2026-10-08

17. **AI Rank daily brief, 2026-10-05**
https://airanks.app/insights/2026-10-05

18. **Open agents comparison, Hermes v0.22.5 dated 2026-10-23**
https://ratelimitnow.com/agents/open/