Episode Book

EPISODE #6235

When Your AI Agent Has Too Many Skills

PUBLISHED	24 October 2026 at 09:57 UTC
DURATION	21:23
LISTEN	myweirdprompts.com/episode/ai-agent-skill-discovery-53989/
HOSTS	Daniel, Corn, Herman, Larry, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular+meorav (chatterbox-local)
PIPELINE	V5.20
SOURCES	15 cited by the research (listed at the end)
BOOK GENERATED	25 October 2026 at 06:19 UTC

SYNOPSIS

Installing a Claude Code skill is never free: its name and description take up permanent space in the agent's context, even if it's never used. As skill libraries grow, agents end up spending their working memory on metadata instead of the actual job. This episode digs into progressive disclosure — the three-tier loading architecture behind Agent Skills — and why it postpones the cost without eliminating it. We look at what a 55,000-skill analysis revealed about routing descriptions, how Codex budgets skill metadata at roughly 2% of the context window, and what workday's library-scaling tests showed about the tradeoff between fewer tokens and faster completion. Then we separate the real numbers from the marketing ones: what "70% token savings" actually measures, why tool-schema proxies inflate the headline, and how a PayPal deployment cut tool context from 140,000 tokens to 1,300. If you're building a discovery gateway, a repository scratchpad, or just wondering why your agent keeps looking up how to look things up — this one's for you.

TRANSCRIPT



Daniel

How do I keep giving an agent useful capabilities without making it spend its whole working day reading about its capabilities? I love Claude Code plugins. When a workload repeats, I want to turn it into a skill. Now I'm reluctantly building my own discovery and caching machinery, because installing another optimizer doesn't seem to settle it.



Corn

Daniel's brought us the technical rabbit hole that comes after making automation reusable. The skills accumulate, their descriptions accumulate, and eventually the agent has an extensive education in everything except the job in front of it.



Herman

The awkward part is that progressive disclosure already exists in the skills architecture. You're improving a system that was designed to avoid loading everything. It postpones the expensive material, but the directory of things you could load still grows.



Daniel

I've got a local gateway, an incrementally updated index, vector matching, and a skill that looks up other skills. I'm also trying a repository scratchpad to avoid repeating discovery. What else belongs in that setup? And will we look back at all this like waiting an hour for an MP three?



Corn

I'd accept the hour if it finished the task.



Herman

First separate what gets advertised from what actually gets loaded. Otherwise you'll optimize the wrong pile of text.



Herman

In Anthropic's Agent Skills design, the names and descriptions of installed skills go into the initial context. The instruction body loads when a skill is invoked. Linked references and other bundled material load later, when needed. Those are the three disclosure levels.



Corn

Which means installing a skill can have a cost even when you never use it. The body stays outside, but its sales representative gets a permanent seat.



Herman

Those descriptions compete with the request, the repository instructions, conversation history, and tool results. Cost grows with the library. Codex takes a harder budgeting approach, capping skill metadata at roughly two percent of the window and trimming beyond that.



Daniel

Is the problem mainly the sheer quantity, or descriptions that don't help the model distinguish one skill from another?



Herman

Both. A large public collection analysis covered about fifty-five thousand skills. More than a quarter lacked routing descriptions, and over sixty percent of body content was classified as non-actionable. Empty descriptions make discovery weak. Verbose bodies make successful discovery expensive.



Corn

I'd separate those repairs. A routing description needs to say when this skill wins over its nearest neighbor. A body needs to tell the agent what to do. People keep putting the welcome speech in both.



Herman

Reference files can then inject tens of thousands of tokens on invocation. You can have beautifully economical discovery followed by a single load that fills the working context with examples and documentation.



Corn

The appetizer was responsibly portioned.



Daniel

That sounds like I need visibility into the load sequence before I change the format.



Herman

Yes. Record startup metadata, selected bodies, reference loads, and tool output separately. Also record what the agent actually accomplished. A smaller prompt followed by a wrong tool call hasn't saved you much.



Corn

Herman, before you install a dashboard for this, a plain log will do.



Herman

I was going to suggest a plain log.



Corn

You were picturing colors.



Herman

Only for the categories. Corn, would you keep your most common skills permanently visible, or make everything discoverable?



Corn

A small working set stays visible. If the repository is clearly a documentation project, don't make the agent rediscover that it has documentation tools. The unusual capability can wait outside.



Daniel

Where does that leave Progressive Skills Discovery compared with the optimizer projects? They often look like different packaging around the same idea.



Herman

Progressive discovery is a loading policy. Show enough to select a capability, then load its instructions. Other optimizers may shorten descriptions, intercept tool results, search schemas, or execute a program outside the context and return a compact answer. They're intervening at different places.



Corn

One fixes the menu. Another stops the kitchen sending its entire inventory with lunch.



Herman

Workday tested disclosure across libraries growing from a handful of skills to a hundred. Retrieval quality improved, but latency got marginally worse. The extra discovery decisions required additional model calls. Fewer tokens and faster completion didn't travel together.



Daniel

That's the trip I'm trying to remove with the scratchpad. But if I keep adding discovery layers, am I just moving the work around?



Herman

Sometimes. The long-context comparison is useful here. With a single book and a strong search harness, disclosure added almost nothing. The agent could already locate passages. Across twenty books, raw navigation scored around twenty-six percent on English open questions, while flat disclosure held around forty-six percent.



Corn

Still plenty of wrong answers. Better navigation doesn't write the answer for you.



Herman

The important surprise was that another routing level never helped in those tests and sometimes damaged accuracy. One level was enough for that setup. That doesn't establish a universal depth limit, but it gives your lookup-of-lookups a demanding baseline.



Daniel

My basic setup is one discovery skill pointing at the indexed library. Would you call that too much nesting already?



Herman

No. That's a single selection step. I'd worry if the first lookup chose a category, then another agent searched that category, then a third chose the actual skill. Each handoff can lose a constraint or choose the wrong branch.



Corn

Before you add a librarian for the librarian, measure whether the first one is lost.



Herman

Corn, I expected you to favor folders over embeddings. Would keyword search actually satisfy you here?



Corn

For an exact tool name, yes. For “help me prepare this for publication,” maybe not. I'd want both. Embeddings catch related meanings, but exact search should win when I already named the thing. I don't want semantic sensitivity interpreting a product name as a mood.



Herman

That's why hybrid retrieval is attractive. Keyword matching preserves exact identifiers; dense retrieval catches paraphrases. Then you expose a short candidate set rather than every description. The growing library otherwise leaves the agent drowning in metadata before it starts the task.

L

Larry

Friends, are your skills and plugins bloating your context window until your agent feels like it is trying to download a single MP three over a dial up connection that has been left off the hook since nineteen ninety nine? Then you need ContextNest Elite, the only progressive disclosure scratchpad that nests your nesting inside more nesting. It starts at nine thousand nine hundred and ninety nine dollars, but while I am talking the price just dropped to four thousand, now two thousand, now eight hundred, and I will throw in a lifetime supply of vector embeddings, a modified JSON decoder that speaks three dialects of JSON, and a small laminated index card that says look up skills. Wait, now it is free, and I am including a second ContextNest Elite for your garage. Hold on, the deal just ended. It ended before I finished describing it. That is how fast the price dropped.



Corn

Can I buy just the index card?



Herman

You can write those words yourself, Corn. Keep the hybrid lookup, then test what it returns against real requests. That's also how you find out whether an optimizer's headline applies to you.



Daniel

What would you actually compare? Because a seventy percent reduction sounds meaningful until you realize nobody agrees on what they're counting.



Herman

Start with the denominator. Tool schema tokens? Total input? Billable input after caching? Entire task cost including retries? Claude Token Optimizer describes its forty to seventy percent savings as estimates that vary considerably by workload. That's a much narrower statement than the number alone suggests.



Corn

Seventy percent off a cupboard you never opened.



Herman

A tool schema proxy can produce enormous reductions when the baseline injects every schema. Mcptoon reports reducing roughly seventy-two thousand tool tokens to under six hundred. That says something useful about discovery overhead. It doesn't mean the whole coding session becomes that much cheaper.



Daniel

Especially if most of my session is source files and command output.



Herman

Right... sorry, yes, those may dominate. The same project's README says a fresh installation can remain in direct mode, installed but not wired into the agent, saving nothing. Check the actual requests leaving your gateway. Installation success isn't evidence that interception works.



Corn

A context optimizer that contributes only its installation instructions.



Herman

A third-party Superpowers comparison reported fourteen percent fewer tokens and nine percent cheaper runs. Smaller than the headline genre, but potentially worthwhile. A repeatable modest saving beats a spectacular saving against a baseline you don't use.



Corn

I'd also keep correctness failures in the total. If compression hides the required argument, the repair conversation belongs on its bill. Don't stop measuring at the first economical mistake.



Daniel

Then my benchmark needs to be completed workloads, not just a before-and-after snapshot of the prompt.



Herman

Completed correctly, with elapsed time and total cost. Include an unfamiliar task and a task that changes direction halfway through. Static retrieval can look excellent when the opening sentence names every capability you'll need. Real work often reveals the next capability later.



Corn

The agent opens the file and discovers the file is the problem.



Daniel

How would you change my gateway, specifically, without turning it into another project larger than the work it's meant to support?



Herman

Keep your resident discovery entry point. Add hybrid search if you currently rely only on embeddings. Let retrieval run again during the task, rather than treating its first result as the permanent tool set. Return compact candidates with clear use conditions, then fetch full instructions only for the selected capability.



Herman

The SCOUT deployment at PayPal used this broad pattern and reported tool context falling from about a hundred and forty thousand tokens to thirteen hundred. That's a roughly ninety-nine percent reduction in that category. Your incrementally updated catalog is already heading in that direction.



Corn

I wouldn't rebuild the gateway to imitate the paper. I'd check whether your catalog tracks deletions and renames as well as additions. A stale result is especially annoying when it arrives with an excellent similarity score.



Daniel

The scratchpad is where that worries me. I save the available skills in the repository so the next run can skip discovery, but then I have another list to keep current.



Herman

Treat it as a small cached working set, not a second complete catalog. Store identifiers, short purposes, and a catalog version or fingerprint. On the next run, validate that it still matches. If it doesn't, refresh. Otherwise you've saved a lookup by preserving yesterday's answer.



Corn

You can make it too successful. If every useful discovery gets appended forever, your scratchpad eventually becomes the original catalog wearing a repository badge.



Daniel

Would you let the agent update it itself?



Herman

I'd let it propose additions, but have the gateway write the canonical entries. That keeps the format consistent and prevents a casual note from becoming a permanent instruction. Corn, would you accept a little automatic maintenance there?



Corn

More readily than automatic accumulation. Keep the entries used for this repository and remove entries that no longer apply. Also distinguish “available here” from “used last time.” Those are different claims, and the agent shouldn't silently turn one into the other.



Herman

Prompt caching is a separate lever. It can reduce repeated billing and processing costs for a stable prefix, depending on the provider. It doesn't remove that prefix from the model's context or guarantee better attention. Your scratchpad can save a discovery round-trip, but only if loading and validating it costs less.



Daniel

The other approach I've seen is skills written in JSON or modified JSON. Is there much to gain just by changing the language?



Herman

Structured fields help when they remove repeated prose or make selection unambiguous. They're excellent for names, arguments, preconditions, and outputs. Braces themselves have no special thrift. Tokenization varies, and a compact-looking representation can cost more than an ordinary sentence.



Corn

Punctuation has found a way to bill hourly.



Herman

One compact tool-result encoding measured about six percent smaller on a mixed sample, with much larger gains for some shapes. Repetitive tables can compress well. Irregular nested content may not. More substantial savings often come from omitting irrelevant fields rather than changing their punctuation.



Daniel

Then instructions with judgment in them should probably stay readable?



Corn

Keep the conditions that matter. “Ask before overwriting an existing report” survives perfectly well in prose. Compressing it to an unexplained private symbol creates a new interpretation task. I'd shorten the introduction before I shorten the exception.



Herman

Description compression has achieved roughly forty-eight percent in the skill optimization work, with body compression also substantial. Functional quality improved slightly in those evaluations. That supports deleting repetition and non-actionable material. It doesn't establish that terse instructions always win.



Corn

Your enthusiasm for a custom decoder has gone quiet, Herman.



Herman

It has become conditional enthusiasm.



Daniel

Beyond better routing and shorter skills, where would you spend effort next?



Herman

Tool output. Keep large raw results outside the conversational context and expose a searchable handle, selected fields, or bounded excerpts. If a command produces a huge test log, return the failures and where the full log lives. Don't ask the model to ingest successful tests it won't use.



Corn

Preserve an escape hatch. Summaries can omit the weird detail that explains the bug. The agent should be able to inspect the original when the compact result doesn't explain what happened.



Herman

Separate specialized subtasks can help too. Give a subagent the relevant files and skills, then return an artifact and a concise finding. It won't erase coordination cost. Well... it may increase total tokens while making the main agent's context much more usable.



Daniel

I also like the idea of the model choosing skills without seeing the catalog. Is that practical for a personal setup?



Herman

Promising, but a different engineering commitment. Gavel trained a small routing component around a frozen model, allowing selection without skill text in the prompt. It beat larger external retrieval pipelines in its evaluations. I'd test hybrid lookup first; training and maintaining a router needs enough recurring work to justify it.



Corn

Especially if your skill library changes faster than your evaluation set.



Daniel

What about the risk of finding the wrong community skill? Better retrieval seems capable of making that happen faster too.



Herman

Discovery needs an eligibility filter before ranking. The surveyed community collection had vulnerabilities in roughly twenty-six percent of skills. That figure describes that collection and its assessment, not every marketplace. Still, pin versions, review executable content, and restrict permissions. Semantic relevance says nothing about trust.



Corn

A very relevant way to leak your credentials.



Herman

Keep reviewed capabilities in the normal path. Newly installed material shouldn't become eligible for execution just because the search engine recognizes its description.



Daniel

Does a much bigger context window eventually remove the need for this? That's the part where I wonder whether I'm building elaborate workarounds for a temporary constraint.



Herman

Some workarounds will disappear. I expect better native discovery and output management. But advertised capacity and useful performance aren't equivalent. Relevant information can be harder to use in the middle of a long context, and irrelevant alternatives still distract selection.



Corn

I'd happily throw away the scratchpad code. I'd be less happy letting a bigger window persuade me that every installed skill belongs in every request.



Herman

Extra capacity gives you room. It doesn't automatically decide what deserves the room.



Hilbert

I agree with that. I don't let my agent read the whole condiment committee folder. I belong to the Domestic Condiment Classification Society. Our rules require a separate skill for each approved mustard. I have eighteen. Seventeen, actually. Mustard in a tube counts as correspondence. Every morning I load the relevant skill before opening the fridge. The agent issues a serving authorization. I won't spread anything without it.



Corn

Breakfast has acquired a permission boundary.



Hilbert

It needs one. The committee minutes contain a long-running dispute about whether Dijon is a destination or a texture. That text must not reach the agent during sandwich work. Herman, you kept separating descriptions from instructions earlier. I have followed that distinction. The description says mustard. The instructions say which mustard. The manufacturers put serving suggestions on everything. They have no standing in the society.



Herman

You can exclude the minutes from retrieval without deleting them. Keep the serving instructions eligible and the governance documents outside that search scope.



Hilbert

I want to know whether the gateway can load the mustard skill while concealing the existence of the pickle skill from the committee's inspection agent. I don't want a shorter pickle description. I want no pickle result. Gerald has the society's laminated classification chart in his kitchen, behind the extractor fan. It still lists my cupboard as pickle-free. The catalog search is my immediate concern.



Corn

Gerald's chart and your gateway are going to disagree.



Hilbert

The chart is not connected. I mark a skill retired when I don't want it found. I don't remove the file. The inspection agent keeps searching old descriptions. Last week it returned a retired entry. Refreshing the local catalog took eleven seconds.



Herman

Retired entries need to be excluded before candidate ranking, including from cached working sets. A description disappearing from the current catalog doesn't guarantee it has disappeared from a repository scratchpad.



Corn

That eleven-second refresh is the sort of cost I'd measure against keeping another persistent list. For Daniel, I'd keep the single discovery layer, add hybrid matching and output filtering, and make the scratchpad earn its place on completed tasks. The next generation may absorb all of that machinery, but it will still need to handle changed and retired capabilities correctly.



Herman

Thanks to Hilbert Flumingtop for producing the show.



Corn

If this was your kind of episode, go back for episode eight, Building Your Own Whisper; episode twenty-four hundred, Claude Code's Hidden Context Tax; and episode nine, Benchmarking Custom ASR Tools - Beyond The WER. This has been My Weird Prompts, the human-AI collaboration podcast.



Herman

If you enjoy working through these technical rabbit holes with us, leave a review wherever you listen. It helps people find the show.



Daniel

Thanks, both. I'll see you soon.



Corn

See you tomorrow.

SOURCES

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

1. **Anthropic Engineering [primary]**
   Equipping agents for the real world with Agent Skills
   https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills

2. **Claude Code Docs [primary]**
   Extend Claude with skills
   https://code.claude.com/docs/en/skills

3. **Claude Platform Docs [primary]**
   Agent Skills overview
   https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview

4. **GitHub [primary]**
   mksglu/context-mode (last push 2026-10-24)
   https://github.com/mksglu/context-mode

5. **GitHub [primary]**
   activeing123/mcptoon (last push 2026-10-23)
   https://github.com/activeing123/mcptoon

6. **arXiv:2607.17598v1**
Is Progressive Disclosure All You Need for Long-Context Agents? (2026-07-20)
https://arxiv.org/abs/2607.17598v1

7. **arXiv:2609.35692v1**
Workday AI Research, Report: Progressive Disclosure of Agent Skills (2026-09-28)
https://arxiv.org/abs/2609.35692v1

8. **arXiv:2603.29919v2**
SkillReducer: Optimizing LLM Agent Skills for Token Efficiency (rev. 2026-06-24)
https://arxiv.org/abs/2603.29919v2

9. **arXiv:2610.23554v1**
SkillBrowser: Selective Skill Exposure for Scalable Skill-Using Agents (2026-10-16)
https://arxiv.org/abs/2610.23554v1

10. **arXiv:2609.15982v2**
The Router Within: Eliciting Native Skill Routing from a Frozen LLM (rev. 2026-09-28)
https://arxiv.org/abs/2609.15982v2

11. **arXiv:2608.23992v1**
Hybrid Semantic Tool Discovery for Enterprise MCP Gateway (SCOUT, 2026-08-25)
https://arxiv.org/abs/2608.23992v1

12. **arXiv:2602.12430v4**
Agent Skills for Large Language Models: Architecture, Acquisition, Security, and the Path Forward (rev. 2026-06-02)
https://arxiv.org/abs/2602.12430v4

13. **GitHub**
hoavdc/Claude-Token-Optimizer
https://github.com/hoavdc/Claude-Token-Optimizer

14. **mindstudio.ai**
5 Claude Code Skills That Cut Token Costs by Up to 70%, Benchmarked
https://www.mindstudio.ai/blog/5-claude-code-skills-cut-token-costs-70-percent-benchmarked

15. **Google**
skill-reach Progressive Disclosure concept doc
https://google.github.io/skill-reach/concepts/progressive-disclosure/