Episode Book

EPISODE #6462

When Adding a Reranker Hurts Search

PUBLISHED	5 November 2026 at 13:40 UTC
DURATION	19:40
LISTEN	myweirdprompts.com/episode/reranker-performance-latency-tradeoffs-34332/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	z-ai/glm-5.3
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.26
SOURCES	12 cited by the research (listed at the end)
BOOK GENERATED	6 November 2026 at 11:12 UTC

SYNOPSIS

Cross-encoders are often billed as a painless, universal accuracy boost for RAG pipelines and search boxes. Yet benchmark results range from a fifteen-point lift down to an outright regression, and some experiments show latency exploding two-thousand-fold for no measurable gain. The discrepancy hinges entirely on the retriever's recall curve and token economics. Here is how to diagnose whether an architecture actually benefits from a reranking stage, the trade-offs of self-hosting versus hosted APIs, and how filtering chunks cuts generation costs.

TRANSCRIPT



Corn

The standard story on rerankers is that they're the cheap upgrade every search system should have. Bolt one on, watch the right answers float to the top. That story is half right, and the half it gets wrong costs people months.



Herman

Which is exactly the trap Daniel is poking at this week.



Corn

He's saying: we've done embeddings, we've done reranking, fine, but separately. Take one search tool, a RAG pipeline or even just a website's search box. Version one, no reranker. Version two, reranker bolted on. What does the user actually see differently, and what are the real numbers between them. And the honest answer is the numbers disagree with each other, wildly, and the disagreement is the interesting part.



Herman

I'm Herman Poppleberry, and I've been looking forward to this one, because the two-stage architecture is one of those things where the mechanics are elegant and the marketing around it is terrible.



Corn

Two claims I'll be checking as we go.



Herman

You check. I'll explain. The machinery first, because you can't judge the numbers without it. Stage one is a bi-encoder. It takes the query and turns it into a vector, takes each document and turns it into a vector, and compares vectors with cosine similarity. The trick that makes it fast is that all the document vectors were computed once, at index time, before anybody ever asked anything. A query comes in, you embed just the query, and you're doing a nearest-neighbour lookup against precomputed vectors. Under a millisecond in the typical setup.



Corn

So the documents never meet the query. They meet the query's shadow.



Herman

That's a nice way to put it, actually. The document only ever sees a compressed summary of the query, compared against its own compressed summary. Stage two, the cross-encoder, is where they finally meet. You concatenate the query and the document into a single input and run them through the model together, so self-attention is comparing query tokens against document tokens, word by word. The BAAI model card has the crispest one-line version I know: reranker uses question and document as input and directly outputs similarity instead of embedding.



Corn

The price of the meeting is that you can't precompute any of it.



Herman

Right, that's the whole trade. Every query-document pair is a fresh forward pass through the model. If stage one retrieves a hundred candidates, that's a hundred fresh passes, per query, forever. The bi-encoder did that work once and amortized it across every query you'll ever serve.



Corn

So the practical question is what the user gets for the per-query bill. Daniel asked exactly that, so let's run his two versions.



Herman

Let's do version one first, and I want to steal Qdrant's own tutorial example, because it's a good one. The query is, what is the purpose of feature scaling in machine learning. The corpus contains documents that use the words feature scaling in other senses. And this is the important part of version one: the user doesn't get an empty results page. They get a full, confident-looking list, where the top slots are dominated by lexical and semantic near-misses. Documents that share vocabulary with the question without answering it.



Corn

The plausible wrong thing at the top. Which is worse than nothing, because nothing tells you to keep looking.



Herman

In a website search box you'd call it a bad search engine and retype the query. In a RAG pipeline it's nastier, because the system doesn't show the user the list. It takes the top five chunks and hands them to the language model, which will happily synthesize an answer from whatever it was given. The right chunk sitting at rank seven of ten is invisible to it. NVIDIA's writeup on this is blunt about it: reranking lets you pass fewer chunks with higher precision, because the model at the end only ever sees what you chose to give it.



Corn

And version two?



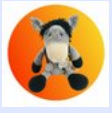
Herman

Version two, the cross-encoder reads the actual question against each near-miss and goes, no, this one's about scaling in the metallurgy sense, this one's about feature scaling in ML, and the correct document moves to position one. The user-visible difference isn't results versus no results. It's plausible wrong thing at the top versus the right thing at the top.



Corn

There's a second difference nobody advertises. Stability.



Herman

Go on.



Corn

The bi-encoder scores each document in isolation, so two near-duplicate documents get near-identical scores, and which one lands above the other is basically coin-flip territory. Rerun the same search, same order isn't guaranteed. The reranker sees the query next to both and can actually separate them.



Herman

That's real, and it's one of the cases where reranking earns its keep honestly. Hard queries where the first stage can't separate near-duplicates. Okay, now the numbers, because this is where Daniel's question gets uncomfortable.



Corn

I was wondering when we'd get to the part where the literature can't agree.



Herman

Start optimistic. The mid-year production survey that gets cited everywhere puts the typical lift at five to fifteen nDCG at ten points across the standard benchmarks, and its latency table has the bi-encoder baseline under one millisecond per query against fifty to five hundred milliseconds for reranking the top hundred. So: meaningful quality jump, for a real but bounded latency bill. That's the version of the story you'll read in most vendor posts.



Corn

Five to fifteen points is the kind of number that sells a component.



Herman

It is. Now the benchmark paper. NVIDIA ran a question-answering retrieval task with a decent embedding model as the baseline, nDCG at ten of about point six one, then stacked different rerankers on top of it. The best one, a Mistral four-billion reranker, took it to about point seven five. Fourteen and a half points. There's your optimistic case, confirmed. But the worst reranker in the same table, ms-marco MiniLM L twelve, scored point five eight. Below the baseline. Adding the component made the system measurably worse.



Corn

Three points worse for adding a part.



Herman

A three-point regression from an upgrade. Same architecture, same corpus, same baseline. The spread from best to worst reranker is entirely a model-choice question, and most of the blog posts about reranking don't mention that there's a downside tail at all.



Corn

Then there's the other end of the spectrum. Hurrell's reranker tax piece from the summer.



Herman

Eoin Hurrell, July. He took BM25 over SciFact, a few thousand scientific abstracts, three hundred queries, and reranked it with a small cross-encoder. BM25 alone: point six six two nDCG at ten, at about a tenth of a millisecond. Rerank at twenty candidates: point six seven one, at two hundred twenty-five milliseconds.



Corn

Two thousand times the latency.



Herman

For a gain of point zero zero nine that doesn't escape the confidence interval. Statistically indistinguishable from doing nothing, at two thousand times the cost.



Corn

So the field's published numbers run from plus fifteen to zero, and the measured ones run to minus three. That's not a spread, that's a disagreement about whether the thing works.



Herman

And both headline numbers are correct. That's the part I want to sit on, because the reconciliation is the single most important thing to understand about rerankers, and almost nobody leads with it.



Corn

Say it, then.



Herman

A reranker reorders what the first stage hands it. It cannot promote a document the retriever never returned. So the reranked score is hard-capped by first-stage recall at whatever k you fed it. Hurrell's own figure on SciFact: recall at five is about point seven two, at ten point seven seven, at twenty point eight two. Only at the top of that climb does the reranker's lift appear. At five and ten candidates, moving essentially zero.



Corn

Wait. So in his experiment the reranker looked useless because the first stage had already found the answer inside the top twenty, and the reranker's whole job is rescuing answers that are further down.



Herman

That's it exactly. His corpus was one where BM25 is unusually strong. The five-to-fifteen-point numbers come from benchmark suites where the first stage is a dense retriever with high recall at a hundred candidates, and there's a lot of room between recall at a hundred and what the user sees at the top five. Same technique, opposite verdicts, because the foundation differed. There's a rule of thumb in the production writeup that compresses the whole diagnostic: the reranker pays off when recall at a hundred is materially above recall at five on your eval set. That gap is the reranker's entire job description.



Corn

So the first question isn't which reranker. It's what your recall curve looks like.



Herman

Diagnose the first stage before you add the smart layer. Hurrell says something close to that verbatim, that the decision gets made as a fix for feelings of poor quality. Add the clever part without measuring the foundation.



Corn

Which is a very human move. The search feels bad, the fix must be the missing clever part.



Herman

Speaking of which, this is where I check the call sheet. Franklin, you're on.
Franklin: Sorry, I've been nodding along for weeks now and I'm fairly sure I've missed step one entirely. Is reranking just, like, when the search box puts the obvious answer first instead of the one with all the right words in it, the way Google guesses what I meant when I typed the wrong thing?



Herman

Close enough. Think of it like a librarian who first grabs every book with the right words on the spine, then actually opens the ones that look promising before handing you one.



Corn

Generous of you, after all that.



Herman

Back to it, then. The recall ceiling reframes everything, because it turns the reranker from an accuracy feature into something more like a cost optimization.



Corn

That's the NVIDIA argument, isn't it.



Herman

It is, and it's the better argument, honestly. Their pitch isn't reranking finds better documents, it's arithmetic. A Llama three point one eight-billion model costs roughly seventy-five times more to process five chunks and generate an answer than the NeMo retriever reranker does per token. So the reranker is seventy-five times cheaper than the generator. Which means you can afford to rerank a wide candidate set, a hundred chunks, and pass fewer chunks to the language model, and the total bill goes down while quality goes up. They report about twenty-one and a half percent cost savings from that rebalancing alone.



Corn

Spend the cheap compute to shrink the input to the expensive compute.



Herman

That's the entire game. The reranker's real job in a production RAG system is being a cheap bouncer that decides what the expensive model gets to see.



Corn

Let's put numbers on the bouncer's salary.



Herman

Per-pair, on a single L40S GPU: the little MiniLM twelve reranker is around four milliseconds a pair, so four hundred milliseconds to do a hundred. The workhorse, BGE Reranker v2 M3, about twelve milliseconds a pair, one point two seconds for a hundred. The big Qwen3 eight-billion reranker, around thirty-eight a pair, nearly four seconds for a hundred candidates. And infrastructure moves the number as much as model size: one independent harness measured the same BGE model self-hosted at around two point four seconds average against Cohere's hosted Rerank three point five at under four hundred milliseconds for the same job. Six times, from deployment choice alone.



Corn

And the hosted bouncer costs a tenth of a cent per search.



Herman

Cohere charges a thousandth of a dollar per search, one query with up to a hundred documents. LiteLLM lists the same at two dollars per thousand queries, as of this week. The self-hosting break-even lands somewhere around a million queries a day. Below roughly ten thousand a day, the API is the sensible default, no GPU operations to babysit.



Corn

So we've got a decision tree. When is the tax worth paying?



Herman

Hurrell's conditions are the ones I'd sign: high-stakes, low-volume retrieval. Hard queries where the first stage can't separate near-duplicates. Subtle ordering mattering enough to justify a listwise reranker. And you pay the tax three ways, latency, cost, and complexity, because now you're running and monitoring a second service that can fail independently of the first.



Corn

Which is the part that never makes it into the architecture diagram.



Herman

Never. One more wrinkle worth naming, because it's where this touches what Daniel actually builds. The adoption picture: a Q1 survey from a reranking vendor put about thirty-six percent of production RAG deployments at no reranker at all. Vendor survey, not audited, but it rings true. Most systems ship the bi-encoder and stop.



Corn

That number I believe, if only because I've used enough of the search boxes this industry produces.



Herman

Here's the second-order part, and I think it's the freshest thing in all of this. Hurrell draws the parallel to agents, and it's a sharp one. Point an AI agent at a weak search tool and you reproduce the reranker mistake one layer up. The agent searches, doesn't find what it needs, reformulates, searches again, loops, burns tokens, and looks productive the whole time. Every rephrased query still runs against the same index, the same embeddings, the same blind spots. His line is that reformulation lets the agent ask differently, not make the retriever return a document it has no path to.



Corn

The recall cap doesn't go away. It just hides under the loop.



Herman

And the loop costs more than the reranker would have. It's the same unmeasured upgrade, paid out in generator tokens instead of milliseconds.



Corn

Herman, before we go on. You've been building out your little retrieval test bench in the spare room.



Herman

It's not little, it's a proper eval harness, and yes, I got the recall curve first this time. Old lesson, learned the hard way.



Hilbert

It was ten. I widened it to fifty.



Corn

Fifty what?



Hilbert

Candidates. Before the reranker touches anything. On the forum I moderate. We had the search the way you're describing, Corn, where the confident wrong thing sits on top. I kept saying the reranker was broken. It wasn't broken. My candidate pool was ten and the good thread was sitting at position twenty-three, and no reranker on earth reaches down to twenty-three out of ten.



Herman

How do you know it was at twenty-three?



Hilbert

Because I score the top ten by hand. Every week, same queries, a set I fixed in two thousand eight. I print the results and I keep them in a binder, and I mark them. Yellow is answers the question, orange is uses the words, green is my own category. I compute my own score off those marks. It's not the standard formula. Mine weights thread actually answers the question above everything, which the standard one does not do, whatever the rest of you are doing with your discounting. Corn, you'd like the binder. You can hold it. I use three highlighters.



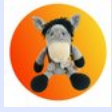
Corn

I have questions about the highlighters.



Hilbert

You can have questions. I don't explain the system, it works. The point is the spreadsheet doesn't lie. Same reranker, same forum, everything the same, and the moment I widened the pool from ten to fifty the right threads started arriving at the top. It was the pool, not the smart layer. Herman's been saying that for ten minutes in fancier words.



Herman

So you've been hand-computing an nDCG variant on a hobby forum since the late two-thousands.



Hilbert

I don't call it that. I call it the standing order. Twenty-three, if anyone wants the number.



Corn

He kept the number.



Herman

Of course he kept the number. That's... actually consistent with everything we've just said, Corn. His binder is a recall-at-k measurement with extra steps. He found the same ceiling, from the other side, with highlighters.



Corn

I want to see the green category.



Herman

Later. Where were we, the agent parallel. The one thing I'd flag from the whole discussion is that reframing. If reranked quality is capped by first-stage recall, and agents are now looping and reformulating on top of the same capped retrieval, the next real improvement isn't a better reranker at all.



Corn

It's a better first stage.



Herman

Or a different answer to where quality actually comes from. And there's a gap in the literature that surprised me: nobody seems to have run a true head-to-head A/B of the same production search tool with and without reranking, measured on user behavior. Click-through, task completion, session length. Every number we quoted today is offline retrieval metrics or latency and cost. The user-perspective half of Daniel's question is inferred from rank order, not measured.



Corn

Six thousand episodes in and the field still hasn't watched a real user type a real query into both versions.



Herman

That's the experiment I'd fund first.



Corn

Then let's leave it there for today. Thanks as always to our producer, Hilbert Flumingtop, holding down the desk and the binder.



Herman

If today's discussion changed how you think about search, the website has the full archive, my weird prompts dot com.



Corn

We'll be back soon. Until then, keep your candidate pool wide.



Herman

And measure the recall curve before you buy the smart layer. See you tomorrow.

SOURCES

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

1. **BAAI model card, created 2024-03-15, last modified 2024-06-24, retrieved 2026-11-05. [primary]**
https://huggingface.co/BAAI/bge-reranker-v2-m3

2. **Qwen model card, created 2025-05-29, last modified 2026-04-16, retrieved 2026-11-05. [primary]**
https://huggingface.co/Qwen/Qwen3-Reranker-0.6B

3. **Cohere pricing page (defines a search as one query with up to 100 documents), retrieved 2026-11-05. [primary]**
https://cohere.com/pricing

4. **Jina model card, created 2026-07-14, retrieved 2026-11-05. [primary]**
https://huggingface.co/jinaai/jina-reranker-v3.5

5. **Eoin Hurrell, The Reranker Tax: When a Smart Layer Can't Save a Weak Foundation, published 2026-07-17, retrieved 2026-11-05.**
https://www.eoinhurrell.com/posts/20260717-the-reranker-tax/

6. **NVIDIA Technical Blog, How Using a Reranking Microservice Can Improve Accuracy and Costs of Information Retrieval, retrieved 2026-11-05. [primary**
vendor]
https://developer.nvidia.com/blog/how-using-a-reranking-microservice-can-improve-accuracy-and-costs-of-information-retrieval/

7. **Moreira et al., Enhancing Q&A Text Retrieval with Ranking Models: Benchmarking, fine-tuning and deploying Rerankers for RAG, arXiv:2409.07691v1, retrieved 2026-11-05 (read pages 4 and 6).**
https://arxiv.org/abs/2409.07691

8. **LLMs Blog, Reranking Engines in Production RAG, mid-2026, retrieved 2026-11-05.**
https://www.llms.blog/posts/reranking-engines-in-production-rag-comparing-bge-reranker-qwen3-reranker-cohere-rerank-jina-colbert-and-flashrank-architecture-latency-quality-and-self-hosted-economics

9. **and https://qdrant.tech/documentation/fastembed/fastembed-rerankers/**
Qdrant docs, retrieved 2026-11-05. [primary, tool docs]
https://qdrant.tech/documentation/search-precision/reranking-semantic-search/

10. **LiteLLM model pricing, prices updated 2026-10-11, retrieved 2026-11-05.**
https://models.litellm.ai/models/rerank-v3.5

11. **OpenRouter, Cohere Rerank v3.5, $0.001/search, 4K context, P50 0.33s / P99 1.67s, released 2026-04-05, retrieved 2026-11-05.**
https://openrouter.ai/cohere/rerank-v3.5

12. **Singh & Jain, ScoreGate: Adaptive Chunk Selection for RAG via Dual-Score Statistical Fusion, submitted 2026-06-12, retrieved 2026-11-05.**
https://arxiv.org/abs/2606.14269v1