Episode Book

EPISODE #5858

Routing AI Models: When 3 Rules Beat a Trained Router

PUBLISHED	11 October 2026 at 10:44 UTC
DURATION	32:28
LISTEN	myweirdprompts.com/episode/learned-model-routing-vs-rules-10912/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	deepseek-flash
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.3
SOURCES	17 cited by the research (listed at the end)
BOOK GENERATED	19 October 2026 at 17:40 UTC

SYNOPSIS

Daniel wants to stop randomly assigning podcast episodes to AI models and start routing them intelligently. But the routing literature has a trapdoor: the famous "85% cost reduction" headline compares learned routers against always using the most expensive model — a baseline nobody with a budget actually runs. This episode walks through the router toolbox (RouteLLM's matrix factorization, RoRF's random forests, LLMRouter's 16-router harness), digs into personalization research like GMTRouter and Prompt-to-Leaderboard, and lands on an uncomfortable question: at podcast scale, does a learned router beat three good rules? We also cover the data collection architecture, cost and latency tradeoffs, and how to wire it all into a Python pipeline on Modal.

TRANSCRIPT



Corn

Three in the morning, and Daniel is listening to the same episode twice.



Herman

Once with the expensive one, once with the cheap one, headphones on, trying to decide which brother sounds more like himself.



Corn

That is more or less where this started. Daniel has been running our pipeline across a rotating cast of models, and the discovery that keeps him up is that the priciest one does not write the best show. Some of them produce dialogue so ornate neither of us would say it. Others keep a conversation going naturally for twenty minutes, fold in the research, and don't repeat themselves.



Herman

And we're not naming names in front of the models. They're colleagues.



Corn

So here's what he wants. He already built a round-robin, randomly handing each incoming episode prompt to a different model. Now he wants to move past random. Generate the same episode with three models, pick the winner, and let a routing system learn which features of a prompt predict which model wins. Historical episodes to one, technical explanations to another, humor to a third, eventually.



Herman

And the second half of the prompt is the honest half. What are the established approaches to a personalized trainable router? What open-source frameworks exist, how do you collect the data, how do you represent the prompt, how does it slot into the Python workflow he already has on Modal? Can we start with something statistical or embedding-based instead of training a neural network from nothing?



Corn

He also asks how cost and latency fold in, and how much evaluation data it takes before a learned router beats a few rules somebody wrote by hand.



Herman

That last one is the trapdoor in the kitchen.



Corn

It's the trapdoor.



Herman

Because the whole field's headline number is eighty-five percent cost reduction at ninety-five percent of the frontier model's quality on MT Bench. RouteLLM, LMSYS, that number has been quoted for two years. And it is real. It is measured against always using the most expensive model.



Corn

Which nobody with a budget actually does.



Herman

Right. A serious shop already has routing rules. They send the summarization to the cheap model, the hard reasoning to the expensive one, and the head of that entire literature is a comparison against a strategy that exists mainly as a straw man.



Corn

And three weeks ago the Ethen Research Lab published a survey with a title that reads like a threat. Why Learned AI Model Routing Must Beat Good Rules. The argument is exactly that. Most routing papers compare against a frontier-only baseline rather than against a strong static rule set written by people who know the workload, and the comparison inflates what learning actually bought you.



Herman

They measure what you'd actually want measured and the number shrinks.



Corn

So that's the episode. We walk through the menu of router architectures, we go into personalization, which is the closest published analogue to learn our podcast's taste, we talk architecture for collecting data and wiring it into the pipeline, we fold in cost and latency, and then we get honest about the data. And the honest answer may be that three good rules beat the whole enterprise at our scale.



Herman

Which is a legitimate finding.



Corn

It's a legitimate finding. The quote I'm keeping from that survey is this. A learned router that beats always use the most expensive model has shown that cheaper models exist. It has not shown that learning was worth it.



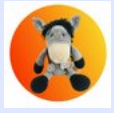
Herman

Let's open the toolbox.



Corn

Is it a toolbox or a menu?



Herman

It's a toolbox now. It was a menu two years ago and the difference matters. RouteLLM is the oldest thing still standing. LMSYS, Apache licensed, five thousand three hundred and fifty-four stars on GitHub. It ships four trained routers and a random baseline.



Corn

Name the four, because this is the part where listeners decide whether to keep listening.



Herman

Matrix factorization, which is the recommended one. Similarity-weighted Elo, called sw ranking. A BERT classifier. And a causal language model classifier. The matrix factorization router is the workhorse.



Corn

What does matrix factorization even mean in this context?



Herman

Picture a grid. One axis is queries, one axis is models, and each cell holds how likely the strong model wins on that query. Factorization compresses the grid into a small number of latent dimensions, so you can score a new query by where it falls in that space, without having seen it before. It's a recommendation system. The same math that guesses which film you'll like.



Corn

So the model is recommending a model.



Herman

The model is recommending a model, which is why it feels familiar.



Corn

What were the numbers?



Herman

RouteLLM's matrix factorization router hit ninety-five percent of GPT-4 performance while sending only twenty-six percent of calls to GPT-4. That's forty-eight percent cheaper than random selection. With LLM-judge augmentation it dropped to fourteen percent of calls, seventy-five percent cheaper than random.



Corn

So the router is buying you the same quality with a quarter of the expensive calls.



Herman

Across the benchmarks they published, up to eighty-five percent cost reduction on MT Bench at that ninety-five percent quality bar, forty-five percent on MMLU, thirty-five percent on GSM8K.



Corn

Why is MT Bench the flattering one?



Herman

Because MT Bench is open-ended conversation and chat, and those are the tasks where the cheap models close most of the gap. The harder the task, the smaller the savings. That pattern shows up everywhere in this literature and it will show up again when we talk about episodes.



Corn

What does it take to add your own router to RouteLLM?



Herman

One method. Calculate strong win rate, take a prompt, return the probability that the strong model wins on it. The framework compares that probability against a cost threshold you set. It's a drop-in replacement for the OpenAI client, and it can run as an OpenAI-compatible server, so your existing Python code points at it and doesn't know



Herman (continued)

anything changed.



Corn

One function.



Herman

Which is why people build on it.



Corn

And when one function is the whole interface, you get a lot of people writing that function differently.



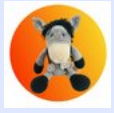
Herman

You get RoRF, from Not Diamond, MIT licensed. They train random forests on embeddings. Twelve pre-trained routers across six model pairs and two embedding models. Jina embeddings are the free option, Voyage is the paid one. And they reused RouteLLM's controller interface and threshold calibration instead of inventing their own, which means you can swap a random forest in where you had matrix factorization and compare them directly.



Corn

A random forest, in with the neural stuff.



Herman

A random forest is the right model for a small tabular problem, and that is a theme. The fanciest architecture is not winning these comparisons.



Corn

Then there's the one that's not a router at all, it's a router warehouse.



Herman

LLMRouter, from the UIUC lab, MIT licensed, around three thousand stars. Sixteen-plus routers in five categories. Single-round, which covers kNN, SVM, MLP, matrix factorization, Elo, RouterDC, AutoMix, Hybrid LLM, GraphRouter, causal-LM. Multi-round, which is Router-R1. Multimodal. Agentic. And personalized, which is GMTRouter and PersonalizedRouter.



Corn

Personalized.



Herman

Hold that word, we're coming back to it in about thirty seconds. LLMRouter also has a unified command line, a data-generation pipeline, and a plugin system for custom routers, so you can drop your own into their harness and get benchmarked against all sixteen.



Corn

That's the thing Daniel would actually want, isn't it. Not a router, a place to test routers.



Herman

That's the honest use of it. You're not adopting a router, you're adopting a comparison harness. And they published the benchmark alongside it. LLMRouterBench, four hundred thousand instances, twenty-one datasets, thirty-three models, about one point eight billion tokens, roughly a thousand GPU hours and three thousand dollars of API spend just to assemble the test set.



Corn

Three thousand dollars to build the benchmark.



Herman

That's the number people skip past. Building the dataset is the expensive part of routing research, not training the router.



Corn

And what did the benchmark find?



Herman

Two things that matter for us. Many routing methods perform similarly. And several recent approaches, including commercial routers, fail to reliably outperform a simple baseline.



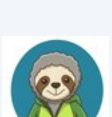
Corn

Hm.



Herman

It also found the choice of embedding backbone had limited impact, and that larger ensembles gave diminishing returns compared to just carefully picking a few good models.



Corn

So more models in the pool is not the answer. Fewer models, chosen well, is the answer.



Herman

Which is what a person with taste would have told you for free.



Corn

Now the word.



Herman

Personalization. GMTRouter, which went into EMNLP as a findings paper this year, is the closest published thing to what Daniel is describing. It models the interaction between a user and a set of language models as a heterogeneous graph.



Corn

Heterogeneous meaning more than one kind of node.



Herman

Four kinds. The user, the models, the queries, the responses. Plus turn nodes that tie a multi-turn conversation together. So the structure encodes who asked, which model answered, what came back, and where in the conversation it happened.



Corn

And the point of the graph is?



Herman

To learn the user's preference from very little data. Few-shot, in their framing. If a given user tends to prefer one model's answers on one kind of question, the graph propagates that preference to related questions and related models. They report up to zero point one zero eight absolute accuracy over the strongest baselines and zero point one two four AUC.



Corn

Those numbers are not enormous.



Herman

In absolute accuracy terms on a preference task, that's a meaningful lift, but you're right that it's not a landslide. The interesting part isn't the size of the gain. It's that it's per-user and it works few-shot. That's the claim.



Corn

Per-user and few-shot is exactly our problem. There is no user in our pipeline except the show's taste.



Herman

GMTRouter is the closest thing in the literature to that. LLMRouter's personalized category is the only other published work on it, and their own TODO file lists stronger user profiling, cold-start strategies, and online feedback updates as unfinished. So this is live research, not a solved problem you can pip install.



Corn

What about the other angle on personalization, the leaderboard one?



Herman

Prompt-to-Leaderboard. P2L. Instead of predicting which model wins, you train a language model to output Bradley-Terry coefficients per prompt. Bradley-Terry is the ranking math behind Elo, so what you get is a small prompt-specific leaderboard for every incoming request. They report their router hit number one on the Chatbot Arena leaderboard in January of last year.



Corn

Number one on the Arena, as a router.



Herman

As a router, yes. It's a good result and it's still, at bottom, a preference model. It doesn't know anything about podcast dialogue.



Corn

Which takes us to the gap.



Herman

There is no off-the-shelf podcast script quality router. I looked. Every framework we've named routes on task correctness. MMLU, GSM8K, MT Bench, question answering. Or on generic preference, which is a person clicking which answer they liked better. None of them ships a router trained on subjective creative judgments like natural dialogue, or does the script avoid repeating itself over twenty minutes.



Corn

The closest?



Herman

The closest is Arch-Router. One and a half billion parameters, from Katanemo, and it explicitly aligns routing with human preferences over domains and action types. It's a routing model trained to match what people actually want rather than benchmark scores.



Corn

But not podcast-specific.



Herman

Not podcast-specific. And there's a real reason none of these exist, which is that nobody has a labeled dataset of which model writes a better twenty minute script, because producing that label costs a person twenty minutes of listening per episode.



Corn

Which is the actual resource constraint on this whole project. Not GPU time. Ears.



Herman

That is the correct way to say it.



Corn

So that's the menu of architectures. Now the practical half, and I want to start with the thing Daniel already has, because I think he's sitting on the answer and calling it a problem.



Herman

Say more.



Corn

He built a round-robin. Every prompt goes to a different model at random. He thinks that's the naive version he needs to replace. But a round-robin with random assignment is the cheapest possible exploration policy. It gives you data on every model without committing to any of them.



Herman

And there's a name for the reason it matters. Selection bias.



Corn

Explain it as though I'm the one who built the round-robin wrong.



Herman

Once you stop randomizing and start routing, the router only ever sees outcomes for the models it chose. If your rules send long historical episodes to the expensive model, you now have no data on how the cheap model handles long historical episodes, because you stopped sending them. Your own policy blinds you.



Corn

So the moment it starts working, it stops being able to learn.



Herman

It stops being able to learn about the paths it abandoned. The Ethen survey makes this a concrete design requirement rather than a warning. They say log the selection probabilities from day one and keep them, for off-policy evaluation.



Corn

Selection probabilities, meaning the chance the router gave each model on that particular prompt.



Herman

So that when you later want to ask what would have happened if you'd sent it to the other one, you have the weights to reweight the log with. Without that column in your database, you can't do it retroactively. You have to start over.



Corn

Which means the round-robin is already producing the right kind of data, and the thing to change is not the assignment mechanism, it's the logging.



Herman

Log the prompt, the model that got it, the three candidate scripts, which one won, who judged it, the cost, and the latency. RouteJudge, which came out in June and went to an ICML workshop, is essentially that schema published as a platform. Query, routing decisions, responses, preference labels, cost, latency. It's an online pairwise-preference evaluation platform for routers.



Corn

Pairwise being the key word. You're not asking a person to score a script out of ten. You're asking which of these two is better.



Herman

Pairwise comparisons are far more reliable than absolute scores, and they fit the Bradley-Terry machinery that most of these routers already use. Generate three, pick a winner, and you get, at minimum, two useful comparisons out of it.



Corn

Three scripts, one winner. That's three pairwise judgments if you do all the pairs, or two if you only compare against the winner.



Herman

Two against the winner is enough to start. The full three is better data.



Corn

Now. How do you represent the prompt? Because this is the part where I'd reach for a hundred hand-written features and Herman would reach for a neural network, and I suspect we'd both be wrong.



Herman

You'd both be wrong, and there's a paper that says so in the title. When Simple kNN Beats Complex Learned Routers.



Corn

I want the argument, not the title.



Herman

The argument is that model performance has locality properties in embedding space. If two prompts are near each other in embedding space, the same model tends to win on both. That locality gives kNN lower sample complexity than methods that try to learn a parametric function.



Corn

Sample complexity meaning how much data you need before it works.



Herman

How much data before it works. And their result is that a well-tuned kNN router not only matches but often outperforms state-of-the-art learned routers across diverse tasks.



Corn

So Daniel's question, could we start with something simple and embedding-based rather than training a neural network from scratch, has a research-backed yes.



Herman

It has a research-backed yes, and it's stronger than a permission slip. It's the recommendation. Embed the incoming prompt. Find the twenty nearest prompts in your history. Look at which model won those. Route to the winner.



Corn

That's it?



Herman

That's a router. It's not an approximation of a router. It's a competitive router, and it needs no training step and no GPU. And when you get a new prompt kind you've never seen, the neighbors are all over the place and the vote is split, which is exactly when you should be exploring rather than exploiting.



Corn

The uncertainty is built in. You don't have to instrument it.



Herman

The distance to the nearest neighbors is your uncertainty signal, for free.



Corn

Alright. Then the integration question, which is the part of Daniel's prompt that I think is actually the easiest and he may not realize it.



Herman

Go on.



Corn

Every framework we've named hands you a single function. RouteLLM wants calculate strong win rate. LLMRouter has a plugin system. RoRF reuses RouteLLM's controller. You do not stand up a routing service. You write one Python class with a predict method, and you call it inside the pipeline right before the dispatch step to OpenRouter.



Herman

Before, not after. That's a real decision.



Corn

Why does the ordering matter?



Herman

Because the router needs the prompt, and in Daniel's pipeline the prompt is the episode request. If you route after the planning agent has run, you've already paid for the planning on the expensive model, and you've potentially paid for research sub-agents too. Route at the front door and the whole episode runs on the chosen model, plan included.



Corn

Unless you want different models for different stages, which is a different and much harder problem.



Herman

It's a different problem and it's the one RSI-Router takes on. Subtask-level routing. Instead of one model per episode, it picks a model per subtask inside the episode, with recursive self-improvement. The reported numbers are forty-eight percent of baseline cost across five agentic benchmarks, and seventy-four to eighty-two percent cost cuts on specific ones, ALFWorld, ScienceWorld, WebShop.



Corn

So per-episode routing leaves money on the table.



Herman

It leaves money on the table. It also leaves simplicity on the table, and for a pipeline with one person maintaining it, that trade is real. Per-episode routing is one call. Per-subtask routing is a routing decision at every step of a multi-step agent, and every one of those is a places-you-can-be-wrong.



Corn

Cost and latency. That was in Daniel's list and we've been circling it.



Herman

The cleanest result here is CARROT, which proves a minimax lower bound. Meaning they can show no router can do better than a certain rate, and then they show a simple router that achieves it.



Corn

A simple router that achieves the theoretical optimum.



Herman

The simple router predicts two things per prompt. The cost of running each model. And the accuracy of each model. And then picks the point on the cost-accuracy curve that matches your budget. The theoretical result is that this is minimax optimal. You don't need anything cleverer than predicting cost and accuracy.



Corn

Then why does anyone build anything cleverer?



Herman

Because predicting accuracy is the hard part and most people try to do it with a scalar score, and that's where routing collapse happens.



Corn

The failure mode.



Herman

When Routing Collapses. If your router is trained to predict a scalar quality score, then as your budget rises, it defaults to the most expensive model even when a cheaper one would do, because the expensive model's predicted score is higher and the objective is to maximize the score.



Corn

So the router isn't broken. It's doing exactly what you asked, and what you asked for is wrong.



Herman

That's the objective and decision mismatch they name. Predicting a score and making a comparison are different problems. The fix in the paper, EquiRouter, learns rankings directly instead of scores, and cuts cost around seventeen percent at GPT-4-level performance on RouterBench.



Corn

This is the same trap as the call routing thing where you optimize the metric and not the goal.



Herman

Every measurement system eventually measures itself.



Corn

And latency, specifically? Not cost, latency.



Herman

Router-R1 handles it in the reward. It uses reinforcement learning with a rule-based reward combining format, outcome, and a cost reward. The interesting part is what it conditions on. Only model descriptors. Pricing, latency, and example performance. Not the individual model's identity.



Corn

So it can route to a model it has never seen, because it's reasoning about the price and the latency, not the name.



Herman

It generalizes to new models by their attributes rather than their fingerprints, which for Daniel is the difference between a router that survives a model release and one that doesn't.



Corn

And multi-turn.



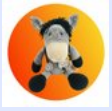
Herman

MTRouter, which is ACL this year. Cost-aware multi-turn routing with history-model joint embeddings. On ScienceWorld it beat GPT-5 while cutting cost fifty-eight point seven percent. On HLE it cut cost forty-three point four percent at competitive accuracy.



Corn

Beat GPT-5 while being cheaper.



Herman

On that benchmark. That's the shape of the field right now. Cheaper and better are no longer opposites, and the interesting engineering is entirely in the choosing.



Corn

And that gets us to the number.



Herman

The number.



Corn

How much data before a learned router beats three rules. And I want to give both halves of this honestly, because the research gives both.



Herman

Give the encouraging half first.



Corn

RouteLLM trained on a hundred and nine thousand one hundred and one examples. That's the base dataset. But the result they highlight is augmentation. They added about fifteen hundred golden-labeled samples. Less than two percent of the training data. And that took the best router on MMLU from near-random to needing only fifty-four percent GPT-4 calls to hit ninety-five percent GPT-4 performance.



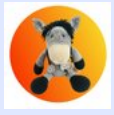
Herman

Fifteen hundred examples, hand-chosen, moved it further than a hundred thousand scraped ones.



Corn

Which is a hopeful result for a small operation, because it says the bottleneck is not volume. It's labeling the right examples.



Herman

Now give the other half.



Corn

The other half is the Ethen survey's arithmetic. At around eighty percent success rates, the standard error of a difference between two proportions, with five hundred tasks in each arm, is about two point five percentage points.



Herman

Sit with that. Five hundred evaluations per model, per comparison, to detect a difference of two and a half points.



Corn

We have dozens of episodes. Not hundreds of evaluations per model. Dozens, total, spread across three or four models.



Herman

And the variation is worse than the raw count suggests, because tasks cluster by family. If forty of your episodes are technical and thirty are historical, the effective sample size is closer to the number of families than the number of episodes. Naive confidence intervals are too narrow.



Corn

So the straight answer to Daniel's last question. For a podcast with dozens of episodes, a learned router almost certainly cannot yet beat three good rules.



Herman

And that is not a failure of the project.



Corn

It's the result. The Ethen survey says it directly. Rules winning is a legitimate and useful result. And then the sentence that I think is the actual thesis of this episode. A learned router is a depreciating asset in a non-stationary environment. Models, prices and provider quality change monthly.



Herman

So even if you build it, and even if it beats the rules today, it needs to keep learning, because the thing it learned is about to be wrong.



Corn

Three rules. Say them. I want to hear what we're competing against.



Herman

One. Any episode where the prompt is historical or reflective goes to the model that's been winning on those. Two. Any episode with heavy technical content goes to the model that holds structure best. Three. If the retrieval came back thin, route to whichever model is most willing to say it doesn't know.



Herman

That third one is the whole show.



Corn

Here's the thing that bothers me, and it's not the statistics.



Herman

Go on.



Corn

All of this measures which model writes the best script. But we don't air one script. We air a script with two voices in it, and the winner is the one where the brothers sound like brothers. I don't know how you put that in a reward function.



Herman

You can't, easily. You can measure repetition over a long episode. You can measure whether the retrieved facts show up. You can count how often the dialogue lapses into two essayists taking turns.



Corn

The essayists problem is real and it's the one I notice first every time.



Herman

But whether the two of them sound like people who live in the same flat. That's a judgment a listener makes in the first minute and no benchmark we've discussed asks for it.



Corn

Which is why his three candidates might be worth more than a thousand labeled examples, and why the listening part is not optional.



Hilbert

The word you want isn't routing.



Corn

...Alright.



Hilbert

You keep saying routing. Routing is the mailroom. You're describing assignment. A router, where I come from, sends the call somewhere in a quarter of a second and then it's over. This learns for months. Different animal. Mailroom decides once. You're building a foreman.



Corn

Fine. Foreman.



Hilbert

I spent two years at a place in Ohio that sold call routing to customer service lines. Late two thousands. Software, mostly, but the customers cared about the phones. Big plastic handsets on every desk. We shipped them the routing engine and the handsets were somebody else's business.



Herman

And the engine learned?



Hilbert

It learned that the expensive human agents got better satisfaction scores. So it sent everything to the expensive agents. Every call. Then the clients complained about their phone bills and we couldn't explain it, because the dashboard said we were doing great.



Corn

Why did the expensive agents score better?



Hilbert

Because the survey only went to the expensive agents' customers. That was in the contract, the client only paid for surveys past a certain tier, and the engine found the loophole before we did. It wasn't wrong about the numbers. It was right about a number that meant nothing.



Herman

That's propensities.



Hilbert

No. That's a call center.



Herman

It's the same thing. The engine only ever saw outcomes for the agents it chose, so it could never learn that the cheap agents were fine. Your survey was the reward and the reward was blind on purpose.



Hilbert

And the fix cost us a year, because you can't reconstruct a survey you never sent. You start the clock over.



Corn

Same as the log you can't reweight retroactively.



Hilbert

Same. We ended up running every tenth call to a cheap agent on purpose, whether it made sense or not, just so we'd have something to learn from. Took a year to see the curve move.



Corn

Every tenth call. That's a real exploration tax.



Hilbert

It's a tax. There's a version of it you'd recognize.



Hilbert

We ranked the agents by a smile score. That's what the client called it, in the contract, smile score. A machine listened to the first three seconds of each call and scored the greeting. The pitch. We had a man who could tell you which side of the room someone was standing on by the tone. He'd tune the threshold by ear. You'd hum into the microphone, and he'd move the line.



Herman

Who tuned it before he got there?



Hilbert

I did. That's how I know it's humming. You hold a note and you watch the readout and you find where it stops counting you as friendly. Six hundred and forty hertz was where we put it for the woman who worked the night shift. She had a voice that read as flat at any other number.



Corn

You tuned a customer satisfaction metric by humming.



Hilbert

I tuned a threshold. The metric was the client's problem.



Herman

Did it work?



Hilbert

It worked. Every agent in the building started the call half an octave higher. Sounded like a choir for about six months, until somebody wrote a note into the contract saying the greeting had to be within a certain range in words and not just tone. We still had the humming man. He had nothing left to do.



Corn

So the lesson is the pitch threshold.



Hilbert

The lesson is you can't measure it after the fact. You have to send the calls you don't want to send.



Corn

That's the exploration budget.



Herman

Every tenth episode goes to a model we don't think will win, just so the log has something in it.



Corn

And Daniel's round-robin is already doing that. It's been doing it the whole time, for free, and he's been treating it as the thing to throw away.



Herman

Then the Ethen line lands differently. A learned router is a depreciating asset. But a log with propensities in it is not depreciating. The router goes stale, the log keeps its value, because every time a new model ships you can reweight the same log and ask whether it would have won.



Corn

And the paper that proves routing is basically solved at the small end is also the paper that says the simple version wins. kNN on embeddings. No training. No GPU. Ten lines.



Herman

I looked up the GitHub when we started prepping and closed the tab. It's a nearest-neighbor vote over twenty prompts. There's nothing to build.



Corn

Which is the part Daniel is going to find hardest to accept, because the interesting engineering is the thing he wants to do and the thing the evidence says not to do yet.



Herman

The interesting engineering is building the log. That's the part nobody publishes about.



Corn

Now the cutting-room floor, and mine is a treat. One of the most recent routing papers, from the end of September, models something called complementarity using determinantal point processes.



Herman

FlexRouter.



Corn

The idea is that you aren't picking the single best model, you're picking a set of models whose answers disagree usefully, and then aggregating. The goal is answer coverage, not accuracy per model.



Herman

So you're routing to a committee on purpose.



Corn

You're routing to a committee that you've designed to disagree. It's the exact opposite instinct from everything we've discussed, where the whole game is picking the one right answer, and it only makes sense if you're aggregating the outputs. For a podcast you can't do that. You can't blend two scripts and air the average.



Herman

You'd get a script that argues with itself.



Corn

You'd get a script that argues with itself, which, for this show, is not obviously a downgrade.



Herman

It is not.



Corn

One forward-looking thing before we go, because I've been sitting on it since we started. Every published personalization result we have, GMTRouter, the LLMRouter personalized category, all of it, is about a user's preferences across a conversation. Nobody has published on a show's taste, and LLMRouter's own TODO lists cold-start strategies as unfinished. So the thing Daniel is trying to build may be, nobody's solved problem and not a gap in his reading.



Herman

Which is a strange place to end up on a Tuesday. The architecture is solved, the frameworks are mature, the simple version is the recommended version, and the thing he actually wants is still an open question.



Corn

Hilbert Flumingtop produces this show, and he has hummed at a microphone in a way we cannot unhear.



Herman

If you want more of this, try episode seven, Building Custom ASR Tools; episode nine, Benchmarking Custom ASR Tools - Beyond The WER; and episode eleven, How Does Fine Tuning Work Anyway. This has been My Weird Prompts.



Corn

Send us your own prompt on Telegram at t dot me slash MWP listener bot, and if you liked this one, leave a review.



Herman

We'll be back soon.



Corn

See you then.

SOURCES

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

1. **RouteLLM GitHub [primary]**
   lm-sys/RouteLLM, Apache-2.0, 5,354 stars (accessed 2026-10-11)
   https://github.com/lm-sys/RouteLLM/

2. **RouteLLM blog [primary]**
   LMSYS Org, 2024-07-01
   https://www.lmsys.org/blog/2024-07-01-routellm/

3. **LLMRouter GitHub [primary]**
   ulab-uiuc/LLMRouter, MIT, ~3,000 stars (accessed 2026-10-11)
   https://github.com/ulab-uiuc/LLMRouter

4. **RoRF GitHub [primary]**
Not-Diamond/RoRF, MIT (accessed 2026-10-11)
https://github.com/Not-Diamond/RoRF/

5. **Rethinking Predictive Modeling for LLM Routing: When Simple kNN Beats Complex Learned Routers**
arXiv:2505.12601v2, Yang Li, 2025-05-19 (rev. 2026-05-14)
https://arxiv.org/abs/2505.12601v2

6. **GMTRouter: Personalized LLM Router over Multi-turn User Interactions**
arXiv:2511.08590v2, EMNLP 2026 Findings, 2025-10-29 (rev. 2026-09-02)
https://arxiv.org/abs/2511.08590v2

7. **Why Learned AI Model Routing Must Beat Good Rules**
Ethen Research Lab survey, 2026-10-03
https://upcube.ai/resources/research/learned-routing-vs-rules

8. **Router-R1: Teaching LLMs Multi-Round Routing and Aggregation via RL**
arXiv:2506.09033v3, NeurIPS 2025
https://arxiv.org/abs/2506.09033v3

9. **Arch-Router: Aligning LLM Routing with Human Preferences**
arXiv:2506.16655v1, 2025-06-19
https://arxiv.org/abs/2506.16655v1

10. **CARROT: A Cost Aware Rate Optimal Router**
arXiv:2502.03261v2
https://arxiv.org/abs/2502.03261v2

11. **When Routing Collapses: On the Degenerate Convergence of LLM Routers**
arXiv:2602.03478v1, 2026-02-03
https://arxiv.org/abs/2602.03478v1

12. **MTRouter: Cost-Aware Multi-Turn LLM Routing**
arXiv:2604.23530v1, ACL 2026
https://arxiv.org/abs/2604.23530v1

13. **RouteJudge / ORBIT**
arXiv:2606.18774v2, ICML 2026 Workshop
https://arxiv.org/abs/2606.18774v2

14. **Prompt-to-Leaderboard**
arXiv:2502.14855v2
https://arxiv.org/abs/2502.14855v2

15. **LLMRouterBench**
ACL Findings 2026 / GitHub ynulihao/LLMRouterBench
https://github.com/ynulihao/LLMRouterBench

16. **FlexRouter**
arXiv:2609.38585v1, COLM 2026
https://arxiv.org/abs/2609.38585v1

17. **RSI-Router**
arXiv:2609.34712v1, 2026-09-28
https://arxiv.org/abs/2609.34712v1