Episode Book

EPISODE #6306

Training Custom Rerankers for Podcast Search

PUBLISHED	27 October 2026 at 18:35 UTC
DURATION	29:07
LISTEN	myweirdprompts.com/episode/custom-rerankers-archive-search-74056/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.22
SOURCES	21 cited by the research (listed at the end)
BOOK GENERATED	28 October 2026 at 09:53 UTC

SYNOPSIS

Building custom search across a dense podcast archive requires more than off-the-shelf vector retrieval. The real engineering work happens during candidate reranking and data curation. This look at search architecture explores chunking transcripts, navigating the tradeoff between recall ceilings and inference costs, mining hard negatives without accidentally penalizing nuanced matches, and using query-specific rubrics like Doug Turnbull's Bag of Decisions before reaching for a full model fine-tune.

TRANSCRIPT



Corn

The text box is the bit I trust, Herman. It's everything after you press enter. Daniel wants to round off this run on search by building something specifically for our archive, rather than attaching somebody else's search box. Listeners should find episodes, his AI tools should check whether we've covered a theme, and full transcripts should count without every passing mention becoming a recommendation. He wants his own weighting. The part he's particularly interested in is whether people actually train their own rerankers, and what that training job involves.



Herman

They do. Businesses train them, individual developers fine-tune them, and model companies build entire products around them. Usually you're adapting an existing model, not teaching a new neural network language from scratch. The difficult decisions happen before the training process starts.



Corn

Good. I was worried we'd need to raise a child.



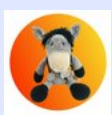
Herman

You need to raise a dataset. Potentially less sleep, depending on the labels. The fork is whether you own that whole relevance pipeline or pay a service like Algolia to operate much of it. Both can give you a polished interface. They're quite different answers to who decides what deserves to appear first.



Corn

Our archive has six thousand one hundred eighty published episodes. That's large enough for me to forget something we've covered, but small enough that a search engine should not get to plead national infrastructure.



Herman

The size is manageable. Transcript structure matters more than the episode count. Index entire episodes as single documents and a short, excellent passage can disappear inside a long conversation. Index tiny fragments and you lose whether we were endorsing an



Herman (continued)

idea, rejecting it, or making a joke about its packaging.



Corn

Including the joke would improve several packaging companies.



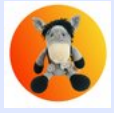
Herman

I'd store episode records alongside transcript chunks, with stable identifiers connecting them. Titles, summaries, topic tags, speakers, timestamps, publication dates. Then use cheap retrieval to collect candidates and a more expensive model to reorder the promising ones. Keyword retrieval, dense embeddings, or both can supply that candidate pool.



Corn

Two different audiences complicate that. A listener asking about sleep might want a coherent episode. Daniel's overlap checker might need one neglected paragraph in an otherwise unrelated conversation. If we optimize them together, the internal tool could decide every episode is about AI because we mentioned a microphone.



Herman

Would you actually maintain separate indexes for those?



Corn

Separate objectives first. Same underlying records, probably. Different retrieval settings, different scoring instructions, different evaluations. I'd rather avoid duplicating the archive just because the consumers have different jobs. The listener gets episode cards. The agent gets passages, timestamps, and enough neighboring text to tell whether it found substance.



Herman

That distinction saves you from training an impossible compromise. Algolia describes query understanding, retrieval, and ranking. The industrial systems often describe a two-stage retrieval and reranking pipeline. Same broad arrangement, different boundaries around the boxes. Before any of those boxes, though, you've got ingestion, corrections, indexing, and metadata that must remain attached to the right audio.



Corn

A search engine can be perfectly relevant to the wrong transcript.



Herman

Yes, and then you need to know which version was indexed. Otherwise the bug report says the quote isn't there, and both the search engine and the listener are correct.



Corn

I think retrieval will be relatively easy here. These transcripts contain the actual language people use, often repeatedly, sometimes because you're reluctant to leave a mechanism alone.



Herman

Some mechanisms deserve a proper goodbye. Still, I wouldn't assume easy until we test it. A listener might search for memory loss when an episode says cognitive impairment. Proper names get mistranscribed. The topic can be present without its usual label. Hybrid retrieval helps because exact words and semantic similarity catch different omissions.



Corn

Fair. Easier than deciding which appearance matters.



Herman

That is where the legal retrieval example is useful. In Beyond Recall Saturation, the larger Qwen embedding model gets roughly ninety-five percent recall with a pool of three hundred candidates on both Chinese legal benchmarks. Expanding the pool near that cutoff buys less than a percentage point. You're spending more to carry increasingly marginal candidates forward.



Corn

Three hundred suspects, five seats in the results.



Herman

The smaller embedder does worse, especially on one benchmark, so model choice still matters. But the curve flattens. Once the right evidence is usually somewhere in the pool, improving its position becomes the valuable work. A reranker cannot recover something retrieval never included.



Corn

I dislike the implication that we can buy a magnificent final judge and still lose the relevant episode at reception.



Herman

You can. Recall sets the ceiling. Reranking determines how much of that ceiling you reach in the results people actually inspect. A bigger pool raises the ceiling, perhaps, but every extra candidate costs scoring time. With long transcript passages, that can be a meaningful serving expense.



Corn

Does that mean we'd rerank passages and then aggregate their scores into episodes? Otherwise one particularly talkative episode could occupy the whole page.



Herman

I'd try that, with a limit on how many passages each episode contributes. Though aggregation needs testing. Taking the maximum score favors one excellent passage. Adding scores favors long episodes. Averaging can punish an episode for including context. There's no innocent arithmetic here.



Corn

The sum rewards me for not ending the conversation.



Herman

An unusual incentive for you. Corn, suppose someone searches for AI regulation.



Corn

For that bare query, yes. For latest AI regulation, no. Recency should respond to intent rather than quietly becoming the answer to every search. I'd label the devoted episode highly relevant, the update useful, and a passing joke about regulators marginal. That's already more information than a binary clicked or ignored.



Herman

Those graded judgments become training material. They also become your test set. Keep those separate, obviously… no, not obviously. It's one of the easiest ways to fool yourself. If closely related queries and duplicate passages land on both sides of the split, you can think you've learned relevance when you've memorized the archive's phrasing.



Corn

You're already reaching for a fine-tune. I can hear the graphics card warming.



Herman

I'm reaching for judgments. You always accuse me one stage early.



Corn

It gives me time to unplug things.



Herman

Here's the part that makes the training job a data job. Random bad examples are usually too easy. Pair an AI regulation query with an archery transcript and the model learns very little. Pair it with an episode about AI company governance and now it has to distinguish subjects that share language but answer different questions.



Corn

Except company governance might actually be a relevant result. Especially if the episode discusses who implements regulation. We would be punishing it for being usefully adjacent.



Herman

That's a false negative. Hard-negative mining finds plausible candidates, but plausible candidates include overlooked positives. You have to review or relabel them. Otherwise training teaches the model to suppress precisely the nuanced matches we wanted it to discover.



Corn

A difficult example isn't automatically a wrong answer.



Herman

Thakur and colleagues made that painfully concrete in Hard Negatives, Hard Lessons. Removing eight of fifteen datasets from a training collection made it substantially smaller and improved BEIR ranking performance by a point. More supervision had been teaching the wrong distinctions.



Corn

Deleting homework improved the student.



Herman

Relabeling false negatives with an LLM also improved retrieval and reranking across their evaluations. The important part isn't that an LLM sprinkled intelligence on the labels. It's that candidate discovery and candidate judgment are different operations. Your mining system can be useful while its assumed labels are wrong.



Corn

I'd want human review concentrated on disagreements. If the model and the labeling assistant both think the sleep episode is about sleep, fine. If one calls an episode central and the other says irrelevant, that's where Daniel's notion of overlap needs to become explicit.



Herman

Yes, although agreement can hide shared mistakes. Keep a random audit sample too. Otherwise two similar models can reassure each other all afternoon. What would you consider overlap, personally? Same topic, same conclusion, or same explanation?



Corn

Same useful work for the listener. Another episode about batteries isn't redundant if it answers a different problem. Another episode explaining why cold weather reduces battery performance probably is, even if the title is new. The overlap checker should return evidence and reasons, not a solemn novelty score.



Herman

That gives us a much better label than same topic. It also suggests an untrained baseline. Generate a little query-specific rubric. Does this passage explain the requested mechanism? Is the topic central? Does it answer the same practical question? Score those decisions rather than asking an LLM one vague relevance question.



Corn

Provided we don't ask six questions that all mean contains batteries.



Herman

Doug Turnbull calls the approach Bag of Decisions. It adds the model's probabilities of answering yes to a set of query-specific checks. On the Wayfair product benchmark, mean ranking quality went from about point five four with BM25 to point six one with the rubric approach. It also beat the plain LLM reranker there.



Corn

More precise instructions beat a larger air of authority.



Herman

It's an experiment worth running before training. Costs move into inference, though. You're generating criteria and evaluating candidates against them. A fine-tuned smaller model may be cheaper when the same judgment must happen continuously.



Corn

If you were building this yourself, what would you have to get right first? I'd still say the definition of a good match, before choosing the model.
Ivan: Lighthouse Six, calling. Groups as follows: seven four, nine two, seven four. Repeat, seven four. The kettle is on the third landing but the lamp is not for you. The searchlight counts the stairs, not the ships. Confirm the spare wick is under the good bowl. Nightjar.



Herman

Seven... four. Confirmed.



Corn

What on earth just came through our search interface?



Herman

The definition of a good match has to become examples. Start with real queries and known useful passages. Then retrieve candidate alternatives, review the close ones, and construct query, positive, negative training examples. Or graded lists, depending on the loss you're using. That's the first actual artifact of the training job.



Corn

Not a folder called training data containing every transcript.



Herman

That folder would teach domain language, perhaps. It doesn't tell a reranker what should outrank what. In the legal example, they used about nine thousand two hundred internal query and evidence pairs, plus roughly twelve hundred public training pairs. That's a curated task dataset, not the whole internet with a legal cover sheet.



Corn

Small enough to inspect badly, large enough to regret doing so.



Herman

Their candidate mining budget was three hundred per query. They began with negatives found by an embedding model, trained, then used the reranker itself to find cases it still confused. That embedding-to-self-mined progression gave the best balance. Extra dynamic refresh stages didn't improve every held-out measure.



Corn

Meaning we shouldn't build an endlessly churning negative factory because the architecture diagram looks industrious.



Herman

Repeated refresh can change what the model is learning against. You want more informative mistakes, but you also want an experiment whose changes you understand. Freeze the dataset version. Record the model, loss, sampling policy, and evaluation set. Otherwise a better score could be a different exam.



Corn

There's also a distinction between episode training and passage training. If the positive is an entire transcript, truncation may remove the relevant section. The model could learn that the right episode is wrong because the evidence was outside its input window.



Herman

That's exactly the sort of unglamorous engineering that dominates. Match training inputs to serving inputs. Preserve enough context, control length, and don't train on beautiful summaries if production will score messy transcript chunks. Summaries may be useful features, but they can make your benchmark unrealistically easy.



Corn

How much machinery are we actually hiring when we say training? A rented accelerator for an afternoon, or a small municipal power station?



Herman

I can't give you an honest duration without the base model, context length, pair count, and training method. A modest encoder fine-tune and distributed training of a large model are very different bills. You need a pilot run that measures throughput and memory use. Then estimate training and serving separately.



Corn

Serving is the bill that keeps its subscription.



Herman

Right. Well... the recurring bill. One developer's task comparison had a fine-tuned encoder with one hundred ten million parameters outperform a seven-billion-parameter generative reranker. The smaller system got roughly seventy-seven percent top-choice accuracy, against seventy-two percent for the larger reranker. That's one task, not a universal ranking of models.



Corn

The larger model was more expensive at being wrong.



Herman

It demonstrates why scale alone is a poor selection rule. A bi-encoder and a generative reranker have different operating characteristics, too. Test the whole pipeline on the task you care about. The legal team's trained larger reranker was preferred in about seventy-two percent of non-tied comparisons on their operational query evaluation. Their result came from task adaptation, not just a bigger download.



Corn

I'd resist training until our baseline makes consistent, describable mistakes. If it misses all exact episode titles, fix lexical search. If it returns twelve chunks from one episode, fix grouping. A neural network shouldn't receive a maintenance ticket intended for a database query.



Herman

You'd make the first version pleasantly boring.



Corn

You say that like you won't immediately replace it.



Herman

I'd preserve it as the control. Probably. A practitioner describing a Claude Code assisted fine-tune put the division of labor well: they chose the model, loss function, dataset, and evaluations. The assistant implemented the fitting work. The thinking wasn't outsourced simply because the keyboard work was.



Corn

The friendly AI down the road can write the trainer. It doesn't automatically know whether our fifth episode about indexing was sufficiently different from our fourth.



Herman

Another training team described distributed embedder and reranker training as deeply expensive, especially if you've asked the model the wrong question. Automation makes executing your objective easier. It also makes executing a bad objective very efficient.



Corn

I'd like inefficiency available as a warning light.



Herman

There's a middle path in ZeroEntropy's training recipe. They collected soft pairwise preferences from an ensemble of LLMs. Which candidate is better for this query, and by how much? They fit an Elo-style Bradley-Terry model to turn those comparisons into document scores, then corrected bias to make scores comparable across queries.



Corn

Instead of pretending each judge hands out perfectly calibrated marks.



Herman

Yes. The calibration step uses maximum likelihood estimation. They released a model under Apache two point zero, so you can inspect and use a trained artifact without becoming responsible for reproducing the entire labeling operation. Owning deployment



Herman (continued)

doesn't require owning every stage of training.



Corn

That's the option I think people skip. Buy no hosted search service, train no new model, still have your own search engine.



Herman

Would you choose that for Daniel's first version?



Corn

I would. Existing embedding model, existing reranker, hybrid candidates, proper episode grouping, and a small judgment set. Keep the scoring interface replaceable. Then train only if the errors justify it. But I'd choose managed hosting if maintaining availability takes him away from the actual product. Independence shouldn't mean personally receiving every midnight alert.



Herman

That last distinction is what Algolia sells. NeuralSearch combines keyword and vector search, merges the results, and ranks them. You don't run the indexes, inference integration, and relevance-learning infrastructure yourself. You still have to supply sensible records and decide what successful search means.



Corn

Their servers cannot discover which timestamp we forgot to attach.



Herman

Their training arrangement also deserves precise wording. NeuralSearch learns from anonymized, combined usage patterns across connected apps. It's pooled learning, not a private bespoke reranker training job you receive afterward. Their documentation says apps in an organization participate by default. Opting out excludes that app's data from



Herman (continued)

future training cycles and forfeits incremental improvements from shared training.



Corn

Own events going in, shared model improvements coming back. That's different from training my model on my archive.



Herman

Click and conversion events help train associations used in ranking. That can be valuable. For our public interface, a useful episode click or a meaningful listening action could be a signal. For Daniel's agent, a click is probably the wrong success measure.



Corn

The agent could click everything and consider itself thorough. You do that with browser tabs.



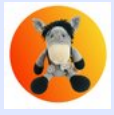
Herman

Those are pending investigations.



Corn

Several have developed civilizations.



Herman

A listener and an agent need different feedback loops. Listener clicks are also biased by position, titles, and familiarity. A result near the top gets attention partly because it was placed there. Managed relevance learning doesn't remove the need to understand what your events actually measure.



Corn

Custom weighting might not need learning at all. If Daniel says central topic beats incidental mention, explicit topic labels could help. If he says exclude episodes already used in this workflow, that's a filter. If he says prefer recent material when the query asks for current tools, that's an intent-sensitive rule.



Herman

Algolia supports custom ranking attributes and boosts for that sort of business control. Dynamic Re-Ranking only reorders records that have passed textual relevance and filters. It can promote a surviving record. It doesn't rescue a record eliminated earlier.



Corn

Reception remains powerful.



Herman

There's also a custom reranking extension for bringing your own machine learning or diversification logic, but it's in closed beta. I wouldn't design a launch around access I hadn't secured. The boundary between managed and DIY is porous, though. You can own labels and evaluations while renting infrastructure, or own infrastructure while using somebody else's model.



Corn

The expensive commitment is whatever becomes hard to replace. Keep the transcripts and judgments exportable. Keep stable identifiers. Have a test suite that can compare the old service with the replacement. Otherwise the migration becomes a debate about whether the results feel worse.



Herman

One small podcast search project has fifty-four hand-labeled evaluation queries. That's more encouraging to me than a spectacular screenshot. Not because that count proves completeness, but because the developer has something repeatable to challenge changes with.



Corn

Fifty-four is a number a person might actually finish.



Herman

The podcast tooling landscape is scattered. There are transcript RAG projects around Lenny's Podcast, speaker-aware BM25 systems, Whisper with pgvector and Ollama, and self-hosted transcription and search projects such as Throughline and Podlog. Useful ingredients, no obvious dominant purpose-built package with a trained podcast relevance model.



Corn

Voilib, the dedicated podcast search engine, is archived. The maintainer pointed to rapid AI advances while stopping active support. General tooling can make a specialized implementation harder to justify, even when a specific archive still needs somebody to define good results.



Herman

The missing piece is often less glamorous than a new retrieval technique. Someone has to decide whether a result answers the request.



Hilbert

Yes, that's the work. I don't care much about the first result. I care about the result just underneath it. Corn, that governance example you caught is where the trouble lives. My shelfback indexes every photograph of a door marked private that I take while travelling. There are eleven thousand six hundred entries. The shelfback sits beside the bread bin, plugged into my router. I use it every evening.



Corn

Private doors make a demanding public collection.



Hilbert

It isn't public. The shelfback must return a door I am allowed to approach without entering. Open doors are negatives. Closed doors with a person holding them are uncertain. A photograph taken from inside is disqualified even if the sign is excellent. Manufacturers keep selling search on the basis that it finds the same word. I've got the same word eleven thousand six hundred times.



Herman

Your shelfback needs the side of the door as metadata. The sign alone cannot establish where the photographer stood.



Hilbert

It has that. It also has a field for whether Marek could carry a folded ironing board through it sideways. I met him travelling through Brno. He had a blue sleeve repaired with yellow thread. I keep thinking about him because he is my fixed width reference. I don't index attractive doors. I index doors Marek could use if he were invited. Herman, you separated retrieval from judgment earlier. That's why the width field isn't allowed to influence the first pass.



Corn

Marek is going to need a correction procedure.



Hilbert

He has one. Every Sunday I send him the disputed shelfbacks. He isn't required to answer. This month he has asked me to stop including revolving doors. The last batch took forty-three minutes to label. There were nine photographs.



Herman

Nine photographs can contain a lot of disagreement. That's the part no training dashboard can turn into an automatic success. You have to distinguish slow annotation from an ambiguous task.



Corn

Forty-three minutes is also more concrete than dataset preparation. For our archive, the difficult labels will be the close thematic overlaps, not the obviously unrelated episodes. A small set can take serious attention if the person judging has to listen around each passage.



Herman

We can reduce that burden without pretending it's gone. Show the query, competing passages, surrounding context, and episode summaries together. Let reviewers mark uncertainty. Keep their explanations. Some uncertainty means the objective needs narrowing; some means there are several good results.



Corn

I'd accept disagreement in the judgments more readily than a model trained to conceal it. The internal tool should say these episodes overlap in mechanism but differ in application. That's useful. A categorical we've covered this could kill a worthwhile prompt.



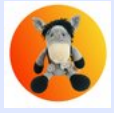
Herman

Evaluation should reflect that. Measure whether the needed evidence appears near the top, but also inspect episode diversity, exact-title retrieval, and central-topic versus passing-mention errors. Latency belongs alongside quality. A model that improves ranking slightly while making every interaction drag may be the wrong choice for listeners.



Corn

The agent can wait longer if it is doing a careful overlap review. A listener typing half a title shouldn't need a committee of language models.



Herman

Separate budgets again. And keep a shadow evaluation when you change models or chunking. Run the same held-out queries before switching traffic. Training isn't the only thing that changes relevance. Transcript corrections, new episodes, and metadata edits can change candidate pools too.



Corn

That's where ownership becomes less romantic. You're not buying one fine-tune and retiring. You're accepting responsibility for the archive as it changes. I still like that control, but I would want the boring operating costs written next to the exciting model choice.



Herman

The idea I'd keep from this is that custom relevance begins with judgments, not custom weights. Rules, an existing reranker, or a fine-tuned model can express those judgments. They are different implementation choices.



Corn

Managed search buys you a pipeline you don't maintain. DIY buys you control over that pipeline, along with the work of proving your changes helped.



Herman

Whether specialized podcast search comes back will depend less on another model release than on someone spending the forty-three minutes to decide what a good result actually means.



Corn

Thanks to Hilbert Flumingtop for producing the show.



Herman

For more along these lines, there's episode eight, Building Your Own Whisper. This has been My Weird Prompts, the human-AI collaboration podcast.



Corn

If you enjoy the show, leave us a review wherever you listen. It helps people find us without training a reranker.



Herman

See you tomorrow.



Corn

Goodbye.

SOURCES

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

1. **Beyond Recall Saturation: Scaling Hard-Negative Reranker Training for Industrial Legal Retrieval, submitted 2026-10-17, revised 2026-10-22 [primary]**
   https://arxiv.org/abs/2610.24478v2

2. **Hard Negatives, Hard Lessons, Thakur et al., EMNLP 2025 Findings, revised 2025-10-18 [primary]**
   https://arxiv.org/abs/2505.16967v2

3. **Algolia docs, Manage model training preferences [primary]**
   https://www.algolia.com/doc/guides/ai-relevance/neuralsearch/model-training

4. **Algolia docs, Get started with NeuralSearch [primary]**
   https://www.algolia.com/doc/guides/ai-relevance/neuralsearch/get-started

5. **Algolia support [primary]**
   https://support.algolia.com/hc/en-us/articles/17700562878481-How-does-NeuralSearch-rank-results

6. **Algolia docs, Dynamic Re-Ranking [primary]**
   https://www.algolia.com/doc/guides/algolia-ai/re-ranking

7. **Algolia docs, custom ranking attributes [primary]**
   https://www.algolia.com/doc/guides/managing-results/must-do/custom-ranking/how-to/configure-custom-ranking

8. **Algolia reranking extension, closed beta [primary]**
   https://github.com/algolia/algoliasearch-extension-reranking

9. **archived podcast search engine [primary]**
   https://github.com/unmonoqueteclea/voilib

10. **Lenny's Podcast RAG, 530 stars [primary]**
    https://github.com/traversaal-ai/lennyhub-rag

11. **BM25 podcast search with 54-query eval suite [primary]**
    https://github.com/derek-rikke/podcast-search-rag

12. **Test-Time Training for Zero-Resource Dense Retrieval Reranking (DART), 2026-05-31**
    https://arxiv.org/abs/2606.01070v1

13. **ConvMemory: A Lightweight Learned Memory Reranker, 2026-05-27**
https://arxiv.org/abs/2605.28062v1

14. **Doug Turnbull, Bag of Decisions Reranker, 2026-10-08**
https://softwaredoug.com/blog/2026/10/08/bag-of-decisions.html

15. **HN discussion of Bag of Decisions, 2026-10-10**
https://news.ycombinator.com/item?id=50034950

16. **and https://news.ycombinator.com/item?id=44582662**
ZeroEntropy zerank-1, 2025-07-16
https://www.zeroentropy.dev/blog/improving-rag-with-elo-scores

17. **Show HN, fine-tuned 110M encoder vs 7B reranker, 2026-09-24**
https://news.ycombinator.com/item?id=49835686

18. **HN comment on finetuning a reranker with Claude Code, 2025-12-30**
https://news.ycombinator.com/item?id=46428658

19. **HN comment on distributed embedder/reranker training, 2026-06-09**
https://news.ycombinator.com/item?id=48469291

20. **HN comment on enterprise search and custom rerankers, 2024-04-04**
https://news.ycombinator.com/item?id=39935321

21. **HN comment on multi-stage retrieval stacks, 2024-03-05**
https://news.ycombinator.com/item?id=39606201