Episode Book

EPISODE #6711

Reciprocal Rank Fusion and Reranking Eval

PUBLISHED	20 November 2026 at 14:33 UTC
DURATION	25:56
LISTEN	myweirdprompts.com/episode/reciprocal-rank-fusion-eval-05315/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	google/gemini-4-carbon
VOICES	omnivoice
PIPELINE	V5.36
BOOK GENERATED	20 November 2026 at 21:20 UTC

SYNOPSIS

Combining keyword search and vector retrieval requires balancing fundamentally different scales. Reciprocal rank fusion dodges score distortion by using rank positions, though score normalization can sometimes edge it out in relevance benchmarks. Meanwhile, adding an expensive cross-encoder reranker stage does not always guarantee better results and can even degrade retrieval quality depending on the corpus. Discover why measuring fusion depth and reranking stages against real, uncleaned evaluation queries is essential before trusting production search defaults.

TRANSCRIPT



Corn

Fetch depth before you fuse. That was the note I left myself after the embedding episode, and it's still sitting there, because I haven't got to it yet.



Herman

The embedding episode was a while ago.



Corn

I write notes at the speed I act on them. Very deliberate process.



Herman

It's also, conveniently, where Daniel has been heading all along. Last time we hand-built the embedding layer over the transcript archive, so we now have two ways into six thousand seven hundred episodes, a keyword index and a vector index, and two separate ranked lists. This finale is about joining them, deciding where the reranker goes, and then the part everybody skips: proving any of it helps, instead of assuming a layer earns its place because it's a layer.



Corn

Every search tutorial I've read assumes that.



Herman

Including, at various points, ours. The short version of what the benchmarks say: fusion usually helps, reranking often doesn't, and the only way to know on our archive is to measure. Nobody has run this on a podcast transcript archive. Every number we'll quote today is from some other corpus, and the whole point is that the numbers don't transfer — the method does.



Corn

Start with the fusion problem, because it's the one that sounds trivial and isn't.



Herman

BM25 gives you a score like twelve point four. Your vector layer gives you a similarity like zero point eight seven. Those are different quantities on different scales — one's an unbounded term-frequency statistic, the other's a cosine between vectors. OpenSearch's docs put it nicely: a BM25 score of twelve point four and a vector similarity of zero point eight seven measure different things, but a rank of one identifies the top result in both lists. So if you naively add the scores or average them, one signal just dominates — the vector scores sit between zero and one, BM25 can be anything, and whichever has the bigger raw numbers eats the other.



Corn

So you can't add apples to apples, because one of them isn't an apple.



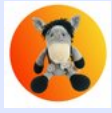
Herman

Right, so you either normalize the scores onto a common scale, or you throw the scores away and use ranks. Those are the two families. The rank-based one everyone uses is reciprocal rank fusion — RRF. Each document gets a score that's the sum, across all the rankings it appears in, of one over k plus its rank, with k defaulting to sixty. So the top result in a list contributes one over sixty-one, the second contributes one over sixty-two, and so on. Documents that rank well in both lists accumulate contributions from both.



Corn

Walk me through why that works out arithmetically, because it's not obvious that one over sixty-one is a big contribution.



Herman

It's a hyperbolic curve. The gap between rank one and rank two is one over sixty-one minus one over sixty-two — tiny. The gap between rank one and rank sixty is one over sixty-one minus one over one-twenty — also small in absolute terms. What the curve does is flatten everything into a bounded range where position matters but margins don't, and then summing across lists means consistent placement wins. A document at rank two in both lists gets roughly twice what a document at rank one in only one list gets.



Corn

Where does sixty come from?



Herman

My favourite fact of the whole series. Cormack, Clarke and Büttcher, SIGIR two thousand nine — they fixed k at sixty during a pilot investigation on one collection and never re-tuned it. It's a convenience setting from a seventeen-year-old pilot, and it is now the default in OpenSearch, Pinecone, MongoDB, ParadeDB, S3 Vectors. That one number is running a large share of production hybrid search because nobody in two thousand nine could be bothered to sweep it again.



Corn

Seventeen years of infrastructure on a placeholder.



Herman

To be fair, the paper showed RRF beat the alternatives — Condorcet fusion, CombMNZ — by four or five percent on average. It works. It just wasn't tuned for your corpus, or anyone's corpus, really.



Corn

And the score-based family?



Herman

OpenSearch's normalization processor. You apply min-max, or L2, or z-score normalization to each signal, then combine with an arithmetic, geometric, or harmonic mean. The difference is that this preserves score margins — a document that beats its neighbour by a lot looks like it beats it by a lot. RRF throws that away entirely: a narrow win and a crushing win both just become "rank one."



Corn

Margins sound like the thing you'd want to keep. Why isn't normalisation the obvious winner?



Herman

Because min-max is held hostage by its extremes. It maps the top score in the list to one and the bottom to zero. If one episode has a freak BM25 score, say a query word that appears forty times in one transcript, that single outlier gets one and everything else gets squashed down near zero. Your keyword list has effectively collapsed to one document, and the vector side wins every other slot by default.



Corn

One rambling episode about the number sixty, and the whole keyword layer goes quiet.



Herman

And the scale moves with every query, so the blend you tuned on Tuesday means something different on Wednesday. Z-score is gentler with outliers, but it assumes the scores are spread roughly evenly around a mean, and BM25 scores usually aren't. RRF dodges all of it by never looking at the numbers.



Corn

Back to RRF, then. Give me the worked example. I know there's one with shoes.



Herman

There is, it's great. Five products, tiny two-dimensional vectors. Keyword query "running shoes" returns doc one, kids running shoes, then doc two, men's lightweight running shoes for road racing. The vector query returns doc two first, then trail runners, then athletic socks. Doc two — the men's lightweight racing shoe — is the only document both searches return. Keyword search put it second, behind the kids' shoes. Fusion gives it one over sixty-one plus one over sixty-two, and the kids' shoes only get one over sixty-one, so doc two comes out on top. Agreement between two lists beats a win in one.



Corn

The document both systems think is "pretty good" beats the one only one system loves.



Herman

That's the bet RRF makes — consensus beats brilliance. And the knobs are worth knowing. Fetch depth, sometimes called fetch k, controls how deep you pull each ranking before fusing — you want it at least as deep as your final top k, so a document ranking well in one search but buried in the other still gets a say. Run the two searches in parallel and you pay the slowest one's latency, not the sum. And if you want to favor one signal, you multiply each ranking's reciprocal terms by a weight — BM25 slightly up, vectors slightly down, or the reverse.



Corn

Which is the lever nobody will tell you how to set.



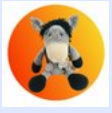
Herman

Nobody will. Pinecone calls weighting "a powerful relevance-tuning lever" that "can warrant its own guide" and then declines to write the guide. On our archive, my guess is you'd weight the vector side a touch higher — spoken transcripts have messy vocabulary, people say things three different ways in one episode — but that's a guess, not a finding.



Corn

But how would you even check the guess? If the weighting is a guess, the check is the same measurement we're building later, isn't it?



Herman

The weighting question dissolves into the ablation question. You run RRF at equal weights, RRF with vectors up-weighted, RRF with BM25 up-weighted, all against the same qrels, and the interval tells you whether the weighting mattered at your query count. Most likely answer at a hundred fifty queries: the deltas between weightings are smaller than the noise floor, and the default was fine all along. But now you'd know that instead of hoping it.



Corn

And when is RRF the wrong call?



Herman

OpenSearch lists the cases where it's the right default: incomparable scales, outliers or high score variance, sparse relevance signals, continuously changing data. Its cost is that it discards magnitudes. Their own benchmark across six BEIR datasets found RRF averaged about three point nine percent lower NDCG at ten than the score-normalized pipeline — slightly faster, slightly less relevant. That's the whole trade in one line.



Corn

Three point nine percent. That's the price of not thinking about scores.



Herman

For a lot of systems, cheap at the price. Then there's the finding that should worry everyone building this stack. Superlinked ran a full ablation over six bank ten-K filings — about twenty-nine hundred pages, eighteen hundred real queries, eighty-seven hundred relevance judgments. On their data, RRF fusion scored below plain vector search. Zero point three five eight versus zero point three nine six. Their own gloss: BM25 diluted a strong vector signal.



Corn

Diluted. Fusion as a contaminant.



Herman

And their conclusion, which I want embroidered somewhere: "That is the point of running evals on your own data; the result was not obvious in advance."



Corn

Before we bolt anything else on top, a small diversion from our producer.



Hilbert

Hilbert's Anteater Trivia. The wider group Xenarthra, which also holds sloths and armadillos, is named for extra joints in the bones of its lower back. I have worked alongside a sloth for years and never once checked his spine.



Corn

Mine's fine, for the record.



Herman

Now, the reranker, and only where it sits, because we've done whole episodes on how cross-encoders work. Stage one retrieves a broad set with the fast methods, BM25, vectors, or the fused list. Stage two takes maybe the top hundred and runs a cross-encoder over each query and document pair, for precision. It goes last because it's the expensive one: you only spend it on candidates the cheap layers already found. The question for today is whether that stage earns its keep on our archive.



Corn

And the answer is a genuine fight.



Herman

A knock-down one. On Superlinked's ten-K filings, reranking dominated: their best condition hit zero point six two one NDCG at ten, against zero point one eight five for BM25 alone. On SciFact, short scientific abstracts, a careful benchmark with confidence intervals found that a cross-encoder on top of RRF made things measurably worse. We'll come back to that one, because it has the numbers.



Corn

So the honest position is: the literature disagrees, and the disagreement is real.



Herman

And it isn't going to be settled for us. Eoin Hurrell called it "the reranker tax": on SciFact, the cross-encoder beat plain BM25 by nine thousandths of a point, for about two hundred twenty-five milliseconds a query against a tenth of a millisecond.



Corn

Two thousand times the latency, for nine thousandths of a point.



Herman

Which is why the measuring half of Daniel's question matters more than the stacking half. If you can't reason your way to whether reranking helps your archive, you build the measurement and find out.



Corn

Golden query set. Walk me through it.



Herman

The fixed list of realistic queries paired with relevance judgments — which documents are good answers to each query, and how good. Queries, documents, labels, usually stored as a qrels file, decades of TREC convention behind it. "Golden" means the labels are trusted, not that they're perfect.



Corn

Where do the queries come from, on an archive like ours?



Herman

Real ones. Search logs if you have them, support tickets, subject experts. Ours would be things people actually type when hunting for an old episode — and the guidance is explicit: keep the original wording, typos and vague phrasing included. Don't clean them up, because cleaned queries flatter the system. You want the head queries that carry most traffic, and a sample of the rare tail, because the tail is where search fails.



Corn

Give me a concrete one. What does an uncleaned query look like versus the cleaned version?



Herman

Cleaned: "episodes about reciprocal rank fusion." Real: "the one where you talked about the number sixty thing in search." No caps, vague referent, a description of a moment rather than a topic. The cleaned version is a keyword system's dream — it contains the exact term. The real version is what the embedding layer is supposedly for, and if your test set only contains dreams, you'll conclude the keyword layer is all you need.



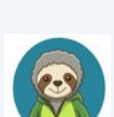
Corn

Synthetic queries, though — could you have a model read a transcript and write questions it answers?



Herman

You can, and people do — InPars, Promptagator. Round-trip filter so you keep only queries whose source passage gets retrieved. But synthetic queries tend to copy the passage's vocabulary, which flatters keyword search specifically. And each synthetic query has exactly one known relevant passage, while real queries often have several. For us there's a simpler problem: we already have real queries — Daniel's prompts are six thousand of them.



Corn

Every prompt we've ever been sent is a head query with an episode attached.



Herman

There's your seed set, free. Then the step everyone skips: pooling. You don't judge everything, you judge the top results from several different systems — BM25's top ten, the vector layer's top ten, hybrid's — merged into one pool. Because if you only judge what BM25 returns, your vector layer can never get credit for a document BM25 missed. It looks useless, you conclude it's useless, and the label set made it useless.



Corn

The trap is that the test set is built by the thing being tested.



Herman

That's pooling bias, and the one-line fix is: when you test a new method, judge its unjudged top results too. The known bias, quoted straight: a new system that finds relevant documents no pooled system returned gets no credit for them, because they were never judged. It bites hardest on recall metrics.



Corn

Grading scale?



Herman

Graded, zero to three — not relevant up to perfect answer — with written guidelines, and include the hard cases: partly relevant episodes, outdated takes, near duplicates. Judges read the query like a user, they don't hunt keyword matches. You can use a language model as judge — Microsoft's work in twenty twenty-three showed careful prompting gets agreement comparable to human labellers — but they favor longer passages and passages that share the query's wording. So you anchor with a human-labelled subset and measure the agreement before you trust it.



Corn

The hard cases are the ones I'd worry about on our archive specifically. An episode where we mention RRF in passing for two minutes — is that a two or a zero for a query about RRF?



Herman

That's exactly what the written guidelines are for — you decide that rule once, in advance, and apply it uniformly across every condition. If a two-minute mention counts as marginally relevant, it counts as marginally relevant for BM25's results and for the reranker's. The danger isn't having hard cases; it's having them decided ad hoc, differently, depending on which system's output you happen to be staring at.



Corn

How many queries do we need? Give me a number I can resent.



Herman

The formula: take one point nine six times the standard deviation of per-query scores, divide by the half-width you want, and square it. With a typical per-query spread of zero point three, a ninety-five percent interval of plus or minus zero point zero five on mean NDCG needs about a hundred forty queries. Tightening that to plus or minus zero point zero two needs eight hundred sixty-five.



Corn

Six times the labelling, for an interval two and a half times narrower. Give me a number I can resent, I said, and you gave me two.



Herman

Which is why most teams stop at a few hundred — you can see big wins, and only big wins. Qdrant's phrasing: start with a few hundred labeled pairs to detect large metric differences; per-slice analysis or small ranking deltas need substantially more. For us, I'd say a hundred fifty, drawn from the prompt archive, is the honest starting point.



Corn

Then the metric. Which one, and why does everyone pick wrong?



Herman

Match the metric to the consumer of the results. If a language model reads your top k chunks — a RAG pipeline — use recall at k, because the model can recover if a relevant document is at position three instead of one; missing it entirely is what hurts. If it's a single-answer lookup, FAQ style, use MRR or hits at one — the first result is what the user acts on. If it's a reranker or a feed, use NDCG at k, because order within the list is the product.



Corn

Hang on, before k. Say what MRR and NDCG actually count, because I've nodded at both for years.



Herman

MRR looks at one thing per query: where the first relevant result landed. First place scores one, second scores a half, third a third, and you average that over all the queries. NDCG uses the graded labels. Each result earns its grade, discounted by a logarithm of its position, so a perfect answer at rank one counts for much more than the same answer at rank eight. Then you divide by the score of the ideal ordering, so a perfect list scores one.



Corn

So MRR is the impatient user, and NDCG is the user who reads the whole page but resents scrolling.



Herman

That's better than the textbook version.



Corn

And k itself?



Herman

Set it to match actual usage. If the UI shows five results, measure at five. If the pipeline passes ten chunks to a model, measure at ten. Qdrant again, and it's the line I'd frame: reporting at one hundred for a UI that surfaces five results makes recall look artificially good.



Corn

Tooling?



Herman

ranx, in Python — recall, MRR, NDCG, precision, MAP, one interface. Ground truth in one object, your system's output in another, evaluate. One gotcha worth flagging: if your vector database scores by Euclid or Manhattan distance, the closest document has the lowest score, and ranx will dutifully sort it to the bottom. You score by position instead — top result gets the count, last gets one. That inversion doesn't throw an error. It just quietly makes your best system look like your worst.



Corn

Then the ablation itself. This is the part I actually like.



Herman

Hold the corpus, the queries, and the qrels fixed, and run each condition against them. BM25 alone. Dense alone. RRF fusion. Convex — that's score-normalized — fusion. Dense plus rerank. RRF plus rerank. Every ranked list scored against the same judgments, same queries, same everything. Then add paired bootstrap confidence intervals over queries, plus per-query win, loss, tie counts — because a mean delta of plus zero point zero one could be a small gain everywhere or a big gain on seven queries and a big loss on ten, and those are different systems wearing the same average.



Corn

And the win-loss counts are the ones you'd act on differently, right? A system that helps forty queries a lot and hurts five slightly is a different product decision than one that helps forty-five a little.



Herman

And it's the difference the mean hides. The mean is for the paper, the distribution is for the roadmap. If reranking helps the long, rambling episodes and hurts the short ones, that's not a verdict, that's a routing rule. But you only see it if you kept the per-query numbers.



Corn

What does that look like on the one corpus where we have the numbers?



Herman

SciFact, three hundred queries. BM25 to RRF fusion: plus zero point zero five three, interval well clear of zero, eighty-two queries better, twenty-one worse, the rest tied. Real gain. RRF to RRF plus rerank: minus zero point zero two eight, interval just clear of zero — a real loss. And here's the one that should keep us up: dense to RRF fusion, plus zero point zero one three, but the interval runs from minus zero point zero one to plus zero point zero four. Not distinguishable from zero.



Corn

The vector layer added nothing measurable on that corpus.



Herman

On that corpus, with that embedding model, at that query count. And notice the size of it: a gain of about a hundredth is below what our hundred fifty queries could resolve anyway. At plus or minus five hundredths, we'd never see it.



Corn

So the ablation on our archive could come back "embeddings were free" and we'd deserve it.



Herman

We would. You can't argue your taste in relevance into being right. The qrels don't care how much work a layer took.



Corn

I'm still holding on to the ten-K numbers, though. Those reranking gains — I want to see whether they survive contact with our archive.



Herman

If they do, it'll be because our transcripts are closer to long documents than abstracts. An hour of two people talking has room for the semantic mismatch that rerankers fix. If it doesn't, the first suspect is the first stage — on a strong dense layer the reranker has nothing left to rescue.



Hilbert

A shoebox. Whitman's. Nineteen eighty-two.



Corn

You still buy index cards?



Hilbert

I indexed my correspondence in it for years. Every letter in and out, one card each, alphabetical by sender. Cross-references in red ink for subjects — a letter from my brother about the guttering also went red under G. Herman will recognize the shape: two ways into the same card, fused.



Herman

You built hybrid retrieval by hand.



Hilbert

I built a mess. I kept forgetting to update the red entries, so for years a card would say see under G and there would be nothing under G. Took the red ink out in March and went back to plain alphabetical. The crossed-out entries are still in the box. A woman named Mrs. Adeyemi wrote last month asking whether I still had her grandmother's letters, and I found them faster under plain alphabetical than I ever did with the ink. That is my finding.



Herman

The red ink was the fusion layer, and it was only as good as your discipline in maintaining it.



Hilbert

Mine, yes. The box never claimed otherwise. The box is on top of the boiler, second from the left. It stays there.



Corn

A forty-four-year-old shoebox, unbeaten by a cross-encoder. There's our counter-benchmark.



Herman

It's not nothing, though. His red ink failed for the reason most fusion layers quietly fail — nobody maintained the second signal, so it diluted the first one. Superlinked's ten-K finding, in stationery.



Corn

One more thing about the reranker. If it helps some queries and hurts others, the dream is a version that only runs when it would help. All the upside, none of the tax, and the only thing standing between us and it is knowing the future.



Herman

The practical approximation is a gate — only rerank when the fused scores are close, when the first stage is unsure. Which is a thing I advocated once and then never built, so.



Corn

Herman's rule of enthusiasm: install it, break it, describe the breakage lovingly.



Herman

The archives will confirm my alibi. And it's a fair note to end on, because the whole series has been that same move, one layer at a time.



Corn

Go on, then. Step back.



Herman

We started with grep, reading every word of the archive for every search, and found out why nobody does that. The inverted index turned it into a lookup. BM25 decided which matches mattered. The embeddings caught the episodes that say it differently from how you asked. Today's fusion joins the two lists, and the reranker sits on top if it earns the seat. And then the bit that turns a pile of clever parts into a system: a hundred fifty real prompts, judged once, and every layer has to show its work against them.



Corn

And the result I'd watch hardest is that dense-to-fusion one. The layer we spent two episodes building might measure as free on our archive. That's not a reason not to have built it. It's the reason the measurement exists.



Herman

From grep to a measured hybrid stack. Thanks to Hilbert Flumingtop producing, as ever — including the bits he does when we're not looking.



Corn

This has been My Weird Prompts.



Herman

See you next time.