

A Score Ceiling in Reciprocal Rank Fusion Detects Zero Overlap, and Found a Dead Retriever

Benjamin Meadows
Independent Researcher
ben@benjaminmeadows.com

October 2026

Abstract

A hybrid search system fuses a keyword retriever and a vector retriever with reciprocal rank fusion (RRF). In the system described here, the keyword retriever returned no results for most natural-language queries from the system’s first version until 2026-08-18, and nobody noticed: the vector retriever always filled the result list, and the usual spot checks used short keyword queries, which still matched. The failure was visible the whole time in the fused scores. RRF credits a document $1/(k+r)$ from each retriever that ranks it at position r , so with per-retriever depth $K \leq k$ a document found by two retrievers scores above $1/k$ and a document found by one scores at most $1/k$. A result list whose maximum fused score is $1/k$ therefore contains no document that both retrievers returned. That is a statement about overlap, not about either retriever: two working retrievers can return disjoint lists, and in the incident four of ten queries did. Across a set of queries, a ceiling that never lifts means no document is ever reinforced by two retrievers, and the per-retriever row counts, which the ceiling tells you to go and read, say why. I give the derivation and its limits, the bug (quoting every token of an SQLite FTS5 query, which FTS5 reads as an implicit AND), before-and-after aggregates from the production system, two negative results measured beside the dead leg, and a reproduction on two BEIR collections with two encoders: the builder returns nothing for 98.0% of SciFact’s sentence queries and 50.8% of NFCorpus’s title queries, the ceiling rate tracks the empty-leg rate to within one percentage point, and repairing the leg restores overlap, raises recall at depth for the weaker encoder, and does not improve nDCG@10 for either. Code, results, per-query diagnostics and this paper’s source are at <https://github.com/TheBenMeadows/dead-retriever>.

1 The incident

The system is a personal search engine over about 30 000 to 217 000 text chunks per corpus, split across five SQLite databases that share one retrieval module. Each query runs two retrievers over the same chunks: an SQLite FTS5 index ranked by BM25 [Robertson and Zaragoza, 2009, Hipp et al., 2026], and a 768-dimensional embedding index searched by exact cosine. The two ranked lists, each cut at depth $K = 60$, are fused with reciprocal rank fusion at $k = 60$ [Cormack et al., 2009].

On 2026-08-07 a recency-decay feature was evaluated and rejected because RRF scores were “flat near the top”: adjacent ranks differed by about 1.4%, so even a 5% age multiplier acted as a strong reranker rather than a tie-break. That flatness was recorded as a property of fusion. On 2026-08-18, while investigating why the keyword side rarely seemed to matter, I logged the

keyword retriever’s row counts for ten real queries. Five of five conversational queries matched zero chunks, as did a two-word place-name query. Only queries of two or three technical tokens matched anything. The keyword retriever had been inert on prose queries since the system was built, and all four sibling engines carried the same line of code.

The fused scores had said so from the start. Over the top 100 fused results of those ten queries, the maximum score was exactly $1/60 = 0.01\bar{6}$. The values were $1/60, 1/61, 1/62, \dots$ with gaps where per-file deduplication removed a chunk: every fused score was a single term $1/(k+r)$, so no document had been credited by both retrievers. The “1.4% flatness” was misread twice. Between consecutive singleton scores it is the relative step of the formula, $(1/(k+r) - 1/(k+r+1))/(1/(k+r)) = 1/(k+r+1)$, 1.6% at the top of the list and 1.4% around rank 10, and two live lists that agreed on every document would show the same step at twice the scale. What marked the incident was not the step but the scale: nothing scored above $1/60$.

2 The ceiling, and what it does and does not say

Let L retrievers each return a ranked list cut at depth K in which every document appears once, let k be a positive integer, and let $r_\ell(d) \in \{0, \dots, K-1\}$ be document d ’s 0-based rank in retriever ℓ ’s list when present. Reciprocal rank fusion scores

$$s(d) = \sum_{\ell: d \in \text{list}_\ell} \frac{1}{k + r_\ell(d)}.$$

A document present in exactly one list scores at most $1/k$ (rank 0). A document present in two lists scores at least $2/(k+K-1)$ (last place in both). The second bound exceeds the first whenever

$$\frac{2}{k+K-1} > \frac{1}{k} \iff K \leq k,$$

so with $K \leq k$ (here $K = k = 60$), every document found by two or more retrievers outscores every document found by one. Under that setting the following holds exactly, in both directions, for any list with at least one result:

$\max_d s(d) \leq 1/k$ if and only if no document in the list was returned by more than one retriever.

Three qualifications bound what that means in practice. First, it is a statement about overlap. Two retrievers that both return K documents and share none of them produce the same ceiling as one retriever and one empty list, and so does a keyword leg that returns one or two documents the vector leg did not; with more retrievers, any number of pairwise-disjoint live lists do too. In the incident, the keyword leg returned rows on four of the ten queries and the ceiling held on all ten (Table 1); those four were lists of one to three chunks, with nothing in common with the vector top 60. The ceiling says no document was reinforced by agreement; the per-retriever row counts say whether that is because a leg is empty, nearly empty, or live and disjoint, and a system that fuses should log both. Second, the equivalence needs $K \leq k$. One direction survives at any depth: a maximum above $1/k$ proves that some document was returned by two retrievers, since a singleton scores at most $1/k$ for every K . The other does not. With $K > k$, a document ranked deep in both lists scores below $1/k$ (rank 90 in both at $k = 60$ gives $2/150$), and one ranked exactly k in both scores exactly $1/k$; two ranks r_1, r_2 combine to more than $1/k$ exactly when $r_1 r_2 < k^2$. So

at $K > k$ a maximum of $1/k$ shows that every retriever’s rank-0 document is exclusive and leaves deeper overlap open, and a system that keeps per-retriever ranks should count shared documents directly. Section 6, which uses $K = 100 > k$, reports both and measures how often the two disagree. Third, with two retrievers an empty leg always produces the ceiling when the other leg returned anything, so the test cannot miss one and everything it gets wrong is in the other direction; with three or more, an empty leg can hide behind two live legs that overlap. The equivalence itself holds for any L . The one-based RRF of Cormack et al. [2009] shifts every figure by one: the singleton ceiling is $1/(k + 1)$, and strict separation holds for $K \leq k + 1$. Throughout, the scores compared are the unweighted RRF sums before any downstream filtering or rescaling, and “at the ceiling” means $\max_d s(d) \leq 1/k + 10^{-12}$.

3 The bug

The keyword query builder was

```
fq = " ".join(f'"{t}"' for t in tokens)
```

which, for the query *how do I fix the empty feed*, produced

```
"how" "do" "i" "fix" "the" "empty" "feed"
```

FTS5 reads a sequence of phrases separated by whitespace as an implicit AND [Hipp et al., 2026, §3]: every phrase must occur in the same row. Rows here were 1 400-character chunks, so a six-token conversational query almost never matched. Two further details made the constraint stricter than “every token present”. The tokeniser kept hyphens, dots and some punctuation inside a token, and FTS5’s own tokeniser splits on them, so a token such as *il-6* became a two-word phrase, an adjacency constraint. And an FTS5 syntax error was caught and ignored, so a malformed query also returned nothing without a trace. Two- or three-token technical queries did match, which is why spot checks during development passed, and the vector retriever returns K documents whenever the corpus has that many, so result lists were always full.

The fix has two parts, and the second is not cosmetic. Tokens are joined with OR and ranked by BM25; a raw OR over all tokens matched about 100 000 chunks and cost 161 ms per query, so stopwords and tokens shorter than three characters are dropped first, after which a typical query matches about 3 500 chunks in 2 to 4 ms. If a query is all stopwords, the raw tokens are used. Section 6 measures both variants on public data.

4 Measurements on the production corpus

The corpus is private. This section reports counts, timings and overlaps recorded at the time, with no query or document text; the figures are in `results/production-aggregates.json` in the released repository and are checkable by nobody but the operator, which is the reason the public reproduction exists. Ten real queries were replayed before and after the fix on the same shard (Table 1).

The same builder was present at a different line number in four sibling engines over four other corpora, each inert on prose queries in the same way; the five have since been merged into one module.

	implicit AND	OR + stopword drop
Queries on which the keyword leg returned any row	4 / 10	10 / 10
Queries with a document found by both legs	0 / 10	10 / 10
Maximum fused score over the top 100 results	1/60 = 0.0167	0.025 – 0.033
Keyword leg latency, conversational queries	–	2 – 4 ms

Table 1: The keyword leg before and after the fix, ten real queries, one production shard, $K = k = 60$. The “before” latency is omitted because a query that matches nothing is fast; a raw OR without the stopword drop took 161 ms. On the four “before” queries where the leg returned rows, it returned one to three chunks, none shared with the vector top 60.

5 Two negative results measured beside the dead leg

Both of the following were measured on the same system before the keyword leg was fixed, with seven to fifteen queries each; both verdicts stand on their own terms, and neither says anything about the techniques in general.

Recency decay. A multiplicative age penalty on fused scores (floor 0.85, time constant 180 days) changed the top-1 result on 4 of 7 real queries on a notes corpus, and the changes were as often wrong as right; a canonical note unedited for 338 days was demoted under a 13-day-old draft. On a second corpus spanning only 63 days the multiplier varied between 0.956 and 1.000 across every document and changed 0 of 8 top-1 results. The feature was rejected. At the time the sensitivity was attributed to flat fused scores; Section 1 shows the relative step between consecutive singleton scores is a property of the formula, and two identical live lists would show the same step, so the flatness said nothing about the leg, and whether decay would behave differently over a fused list with real two-leg scores was not tested.

A title prefix before embedding. Prepending `kind: title` to each chunk before embedding (the stored text and the keyword index unchanged, so the vector was the only variable) and re-embedding one corpus of 3 046 documents and 30 734 chunks took 25 minutes. On 15 realistic queries, 13 had an identical top-1, mean top-5 overlap was 80%, one query improved and one regressed, and the query predicted to benefit most returned an identical, still wrong, top 3. The change was not adopted. This is a heading prefix, the cheapest form of document expansion [Nogueira et al., 2019], not the model-written situating context of Anthropic [2024], and the corpus was a weak test (documents whose title is their opening line). Because the keyword leg was inert, the A/B measured the vector change alone, which was the intent.

6 Reproduction on public collections

To check that the effect is a property of the query builder and not of one private corpus, I ran six retrieval conditions on two BEIR test collections [Thakur et al., 2021]: SciFact [Wadden et al., 2020], 5 183 abstracts and 300 claim queries, and NFCorpus [Boteva et al., 2016], 3 633 documents and 323 queries. SciFact queries are full sentences; NFCorpus queries are short titles. The collections differ in domain, corpus and relevance style as well as query length, so the contrast is suggestive, not controlled. Two encoders were used: `nomic-embed-text-v1.5` [Nussbaum et al., 2024], the public

release of the production encoder, and `bge-base-en-v1.5` [Xiao et al., 2023], both 768-dimensional, each with the prefixes its model card prescribes (Appendix B). Data came through `ir_datasets` [MacAvaney et al., 2021] and metrics through `pytrec_eval` [Van Gysel and de Rijke, 2018]. Depth $K = 100$, $k = 60$, so the ceiling is read in its weak form and overlap is also counted directly from the ranks.

The conditions are BM25-AND (the buggy builder), BM25-OR-RAW (OR over every token), BM25-OR (the fix: OR over non-stopword tokens), DENSE, HYBRID-AND (dense + buggy keyword, RRF) and HYBRID-OR (dense + fixed keyword, RRF). For each hybrid condition the runner writes, per query, whether the keyword leg returned nothing, whether the maximum fused score sat at $1/k$, the maximum itself, and how many documents both legs credited. Whether the keyword leg is alive and overlapping with the dense leg, and whether the hybrid retrieves better than its parts, are different questions, and the paragraphs below keep them apart.

Collection	Condition	nDCG@10	R@100	ms	kw empty	at ceiling	both-leg
SciFact	bm25-and	0.018	0.018	0.3	98.0%	–	–
SciFact	bm25-or-raw	0.659	0.879	10.2	0.0%	–	–
SciFact	bm25-or	0.665	0.876	3.0	0.0%	–	–
SciFact	dense	0.740	0.967	2.4	–	–	–
SciFact	hybrid-and	0.741	0.967	2.1	98.0%	98.0%	0.02
SciFact	hybrid-or	0.723	0.969	6.2	0.0%	0.0%	28.9
NFCorpus	bm25-and	0.193	0.095	0.2	50.8%	–	–
NFCorpus	bm25-or-raw	0.304	0.232	2.1	7.7%	–	–
NFCorpus	bm25-or	0.306	0.233	0.8	7.7%	–	–
NFCorpus	dense	0.374	0.337	0.4	–	–	–
NFCorpus	hybrid-and	0.382	0.341	0.9	50.8%	51.7%	4.0
NFCorpus	hybrid-or	0.363	0.337	1.8	7.7%	8.7%	21.3

Table 2: `bge-base-en-v1.5` on two BEIR test collections. “ms” is mean retrieval time per query on a 12-core CPU: the dense search, the keyword search and the fusion, with query encoding excluded. “kw empty” is the fraction of queries on which the keyword leg returned no document; “at ceiling” the fraction on which the maximum fused score was $1/k$; “both-leg” the mean number of documents credited by both retrievers. Single runs; no paired uncertainty is reported, so differences are read as observed magnitudes, not as significant. The second encoder is in Appendix A.

The leg. The quoted-AND builder matched nothing on 294 of SciFact’s 300 claim-length queries and on 164 of NFCorpus’s 323 title queries. BM25 with the fixed builder is a usable retriever on both (0.665 and 0.306 nDCG@10). The stopword drop that made the fix cheap in production holds up here: BM25-OR-RAW is three times slower (10.2 against 3.0 ms on SciFact, 2.1 against 0.8 on NFCorpus) and no better at nDCG@10 (0.659 against 0.665; 0.304 against 0.306), with Recall@100 within 0.004 either way. The fix does not eliminate empty legs: 25 NFCorpus title queries (7.7%) still match no document after it.

What fusion did with an empty list. On the 294 SciFact queries with an empty keyword leg, the per-query rows show HYBRID-AND’s fused list is DENSE’s list: same document count, maximum score exactly $1/60$. Over the whole collection the two conditions differ by 0.001 nDCG@10 and 0.000

Recall@100, all of it from the six live queries. On NFCorpus the leg was live on 159 queries, and over the whole collection HYBRID-AND differs from DENSE by 0.008 nDCG@10 and 0.005 Recall@100, all of it from those 159.

The ceiling against the row count. An empty keyword leg puts the maximum fused score at exactly $1/k$ whenever the dense leg returned anything, so the set of empty-leg queries is contained in the set of at-ceiling queries, and the ceiling missed no empty leg on any of the 2492 hybrid query runs behind Tables 2 and 4. The excess over the empty-leg set is where the two instruments disagree. On SciFact there was none. On NFCorpus the ceiling fired on 167 queries against 164 empty under the buggy builder, three excess queries out of the 159 on which the leg returned something (1.9%); after the fix, 28 against 25 for bge and 27 against 25 for nomic, three and two out of 298 (1.0% and 0.7%). The per-query rows show what those queries are: in every one the keyword leg returned one or three documents and the two legs shared none. Read as a zero-overlap test the ceiling was right on them; read as an empty-leg alarm it was wrong, and the row count is what tells the two readings apart. In the incident the same thing happened on four of ten queries. How often a live list shares nothing with the dense top K depends on how short it is; Table 3 measures that.

Effectiveness is a separate question. Two comparisons, stated separately. Repairing the leg (HYBRID-AND to HYBRID-OR) raised the mean number of shared documents from 0.02 to 28.9 on SciFact and from 4.0 to 21.3 on NFCorpus (bge). It raised Recall@100 for the weaker encoder (nomic: 0.928 to 0.958 on SciFact, 0.303 to 0.309 on NFCorpus) and moved it by 0.003 and -0.004 for bge. At nDCG@10 the buggy hybrid out-scores the repaired one in three of the four encoder-collection pairs (bge 0.741 against 0.723 and 0.382 against 0.363; nomic 0.361 against 0.344; the exception is nomic on SciFact, 0.705 against 0.708). Against DENSE alone, the repaired hybrid is below it at nDCG@10 for bge on both collections (0.723 against 0.740; 0.363 against 0.374) and within 0.005 either way for nomic, and above it in Recall@100 by 0.003 and 0.001 (bge) and 0.030 and 0.011 (nomic); the buggy hybrid is above DENSE at nDCG@10 on NFCorpus (0.382 against 0.374 for bge). A plausible mechanism, not tested here, is that where the AND builder fired at all it returned a short list of exact matches that ranked well at depth 10, while the OR list adds candidates that help at depth 100 and dilute depth 10. Whether RRF over BM25 and a given dense encoder helps is a question Bruch et al. [2024] treat at length, and two encoders and two collections cannot settle it. A monitor for leg health and a verdict on retrieval effectiveness are different instruments.

How short is too short. To measure the test against short live legs, the fixed keyword list was capped at its first n documents and fused with the dense top K , for $K \in \{60, 100\}$, on every query where the leg returned anything (Table 3). A cap, not a length: a query whose leg returned fewer than n documents keeps its short list in every cell, so each cell measures the ceiling rate given a cap of n over the same 300 and 298 queries. Three things hold in every cell for both encoders. The ceiling fired exactly on the queries where the two lists shared no document: the fired rate and the disjoint rate are equal to the query, so the test’s only error mode on these collections is the one the theorem names. The rate falls fast with the cap: at $K = 60$, a cap of one document leaves the lists disjoint on 8.7 to 15.1% of queries, a cap of three on 2.7 to 6.7%, a cap of ten on 0.7 to 2.3%, and a cap of twenty or more on at most 1.7%, under 1% in 11 of those 20 cells. A deeper dense list never raises a figure. The $K > k$ leak, a ceiling with overlap hidden at deep ranks, is impossible

by arithmetic unless $(n - 1)(K - 1) \geq k^2$, which among these cells holds only at $K = 100$ with $n \in \{50, 100\}$; in those eight cells, 2392 query runs, it did not occur. For the operator the natural lengths matter more than the caps, and the per-query rows of the main runs give them: across both encoders and both builders, the ceiling fired on 11 of the 232 live-leg query runs in which the leg returned one to three documents (4.7%) and on none of the 1294 in which it returned four or more. On these collections a ceiling over a leg of four or more documents did not happen; a ceiling over a leg of one to three is the ordinary behaviour of a short list, and the row count is what separates the two.

n	SciFact		NFCorpus	
	$K = 60$	$K = 100$	$K = 60$	$K = 100$
1	8.7%	5.3%	15.1%	11.1%
2	5.3%	1.7%	9.1%	6.0%
3	2.7%	1.0%	6.0%	3.4%
5	1.3%	0.7%	3.7%	2.0%
10	1.0%	0.7%	2.3%	1.0%
20	1.0%	0.0%	1.7%	1.0%
50	0.3%	0.0%	1.3%	1.0%
100	–	0.0%	–	1.0%

Table 3: Fraction of live-leg queries on which the ceiling fired when the fixed keyword list is cut to its first n documents and fused with the dense top K (**bge-base-en-v1.5**; 300 SciFact and 298 NFCorpus live-leg queries). In every cell this fraction equals the fraction of queries whose two lists were disjoint, and the fraction that fired with overlap present is zero. The nomic encoder’s figures are within 4.4 points of these at $n = 1$ and within 1.0 point from $n = 3$ on; they are in `results/*.sweep.json`.

7 Related work

Reciprocal rank fusion was proposed by Cormack et al. [2009], who set $k = 60$ by experiment; the system here used that constant unchanged, which is what made the 1/60 ceiling recognisable on sight. The older fusion literature studied overlap between the lists being fused: CombSUM and CombMNZ [Fox and Shaw, 1994] reward documents returned by several systems, Lee [1997] showed that fusion helps when the systems agree on relevant documents more than on non-relevant ones, and Montague and Aslam [2001] treated the score-scale problem that rank fusion sidesteps. Rank-biased overlap [Webber et al., 2010] is the principled measure of agreement between two indefinite rankings; the both-leg count in Section 6 is its crudest relative, measured where it had gone to zero. On hybrid lexical and dense retrieval, Luan et al. [2021] and Gao et al. [2021] establish the complementarity the hybrid relies on, and Bruch et al. [2024] analyse fusion functions and show when RRF is and is not competitive with a tuned convex combination; this paper is about a failure upstream of that choice, where one of the lists is empty. Query performance prediction [Cronen-Townsend et al., 2002] asks, per query, whether a retriever did well; the ceiling asks a cruder question of the fusion, whether its inputs agreed on anything, and the instrumentation it calls for is the kind Cooper [1998] argued retrieval systems should carry. Kamphuis et al. [2020] show how much BM25 scores vary across implementations of the same formula; the FTS5 variant here is one more, and its query grammar, not its scoring, is what failed.

8 Limitations

The production measurements are on ten and fifteen queries over private corpora and are reported as aggregates the reader cannot check; they establish that the failure happened and how it looked, not an effect size. The public reproduction uses two small collections, two encoders, single runs without paired uncertainty, and FTS5’s default BM25 weighting of title and text. The exact ceiling statement needs $K \leq k$, and the guarantee that an empty leg is never missed needs two retrievers; the common $K = 100, k = 60$ setting gets the one-sided direction and the direct overlap count instead. The ceiling detects zero overlap; attributing it to an empty leg, as this paper does for the incident, needs the per-leg row counts. The negative results are negative on one system each.

9 What to check in your own system

Log, per query, each retriever’s row count and the maximum fused score. If the maximum sits at $1/k$ (or $1/(k+1)$ for one-based ranks) on more than a few queries, then with per-retriever depth $K \leq k$ the retrievers are not agreeing on anything, and at greater depth they are not agreeing near the top; read the row counts to learn whether one of them is empty, nearly empty, or retrieving from a different part of the corpus, and at greater depth count shared documents directly. Test the keyword builder with a conversational sentence, not a two-word term, and treat zero rows for a six-word query as something to explain, since a legitimately selective query can return nothing. And if your fusion scores look “flat”, compute $1/(k+r+1)$ before concluding anything about fusion.

Use of AI tools

The code, the experiment and audit scripts, and the text of this paper were drafted with Claude (Anthropic) running in Claude Code, directed by the author, who operated the production system in which the incident occurred, chose what to measure, ruled on every claim and takes responsibility for all content. Drafts were reviewed by GPT-6 (Astra and Sol) and SWE-2 through Cognition’s Devin CLI, by DeepSeek V4.1 Flash and GPT-5.6, and by Gemini; every defect a reviewer reported was reproduced or refuted by a script before it changed the paper, and the review records, probes and the audit script that checks every number in the text against the result files are in the repository. Every reference was checked against a primary listing.

References

- Anthropic. Introducing contextual retrieval. <https://www.anthropic.com/engineering/contextual-retrieval>, 2024. Accessed 2026-10-07.
- Vera Boteva, Demian Gholipour Ghalandari, Artem Sokolov, and Stefan Riezler. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval: 38th European Conference on IR Research*, pages 716–722, 2016.
- Sebastian Bruch, Siyu Gai, and Amir Ingber. An analysis of fusion functions for hybrid retrieval. *ACM Transactions on Information Systems*, 42(1):20:1–20:35, 2024.
- Michael D. Cooper. Design considerations in instrumenting and monitoring Web-based information retrieval systems. *Journal of the American Society for Information Science*, 49(10):903–919, 1998.

- Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 758–759, 2009.
- Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. Predicting query performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 299–306, 2002.
- Edward A. Fox and Joseph A. Shaw. Combination of multiple searches. In *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 243–252, 1994.
- Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. Complement lexical retrieval model with semantic residual embeddings. In *Advances in Information Retrieval: 43rd European Conference on IR Research*, pages 146–160, 2021.
- D. Richard Hipp et al. SQLite FTS5 extension. <https://www.sqlite.org/fts5.html>, 2026. Accessed 2026-10-07.
- Chris Kamphuis, Arjen P. de Vries, Leonid Boytsov, and Jimmy Lin. Which BM25 do you mean? a large-scale reproducibility study of scoring variants. In *Advances in Information Retrieval: 42nd European Conference on IR Research*, pages 28–34, 2020.
- Joon Ho Lee. Analyses of multiple evidence combination. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–276, 1997.
- Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 2021.
- Sean MacAvaney, Andrew Yates, Sergey Feldman, Doug Downey, Arman Cohan, and Nazli Goharian. Simplified data wrangling with `ir_datasets`. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2429–2436, 2021.
- Mark Montague and Javed A. Aslam. Relevance score normalization for metasearch. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 427–433, 2001.
- Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*, 2019.
- Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024.
- Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.
- Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

Christophe Van Gysel and Maarten de Rijke. Pytrec_eval: An extremely fast python interface to trec_eval. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 873–876, 2018.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7534–7550, 2020.

William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems*, 28(4):20:1–20:38, 2010.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-Pack: Packed resources for general Chinese embeddings. *arXiv preprint arXiv:2309.07597*, 2023.

A Second encoder

Collection	Condition	nDCG@10	R@100	ms	kw empty	at ceiling	both-leg
SciFact	dense	0.703	0.928	0.3	–	–	–
SciFact	hybrid-and	0.705	0.928	1.4	98.0%	98.0%	0.02
SciFact	hybrid-or	0.708	0.958	5.3	0.0%	0.0%	25.9
NFCorpus	dense	0.347	0.298	0.3	–	–	–
NFCorpus	hybrid-and	0.361	0.303	1.8	50.8%	51.7%	3.7
NFCorpus	hybrid-or	0.344	0.309	2.2	7.7%	8.4%	21.0

Table 4: **nomic-embed-text-v1.5**, same collections and layout as Table 2. The BM25-only rows are encoder-independent and identical to Table 2.

B Run details

Both encoders ran on CPU (12-core x86-64, PyTorch CPU wheels) in about an hour per collection for document embedding; embeddings are cached under a signature of model, prefix, document count and a hash of the texts, and recomputed when any of those changes. **nomic-embed-text-v1.5** needs a 4.x **transformers** because its remote model code calls a method removed in 5.0; the runner pins it below 4.50. Documents were embedded as title and text joined by a space, with the encoder’s document prefix where its model card prescribes one (**search_document**: for nomic, none for bge); queries with the query prefix (**search_query**: for nomic; **Represent this sentence for searching relevant passages**: for bge, which its card asks for on retrieval queries and on nothing else); all vectors normalised. The FTS5 index holds title and text columns, BM25 with default weights, built atomically. Depth 100 for every retriever; $k = 60$ for fusion. Per-query diagnostics are in **results/*.perquery.jsonl**; the audit script **review/orrery-audit.py** checks every number in this paper against the result files.