The $1/k$ Ceiling in Reciprocal Rank Fusion: A Zero-Overlap Test for Hybrid Retrieval

Benjamin Meadows
Independent Researcher
ben@benjaminmeadows.com
ORCID 0009-0007-9698-0910

November 2026

Abstract

Hybrid search runs a keyword retriever and a dense retriever over the same corpus and fuses their ranked lists, often with reciprocal rank fusion (RRF). If the keyword retriever silently returns nothing, the fused list still looks full, because the dense retriever fills it. This paper shows that, at per-retriever depth $K \leq k$, the maximum fused score reveals whether the two lists shared any document. With ranks starting at zero, RRF gives a document $1/(k + r)$ from each retriever that ranks it at position r. With per-retriever depth $K \leq k$, a document returned by two retrievers always scores above $1/k$, and a document returned by one scores at most $1/k$. So a result list whose maximum fused score is $1/k$, a list *at the ceiling*, contains no document that both retrievers returned. The test measures overlap, not health: two nonempty lists that share nothing produce the same ceiling as an empty list, and per-retriever row counts tell the two cases apart. I describe a production system whose keyword query builder quoted every token of an SQLite FTS5 query, which FTS5 reads as an AND over all of them. Replayed on ten real queries, the builder returned nothing for every conversational one, and the fused list was at the ceiling on all ten. On six BEIR collections with three encoders, across 22 362 hybrid query runs at $K = 100 > k = 60$, the buggy builder returns nothing for 50.8 to 98.0% of queries on five collections, and the ceiling missed none of those empty lists, as the derivation requires with two retrievers and a dense list that was never empty. Across both builders, the fused list was also at the ceiling in 174 query runs where the keyword list was nonempty but shared nothing with the dense list, mostly short lists on the larger corpora, and in two runs under the repaired builder where the lists shared one document at deep ranks, the case $K > k$ leaves open. On the sixth collection, ArguAna, each query is itself a corpus document, so the buggy keyword list is never empty and almost never relevant, and the ceiling never appears. Repairing the keyword retriever raises overlap in every collection–encoder pair. At nDCG@10 the repaired hybrid beats the dense retriever alone in 5 of 18 pairs and loses to it in 13; two of the gains and five of the losses are significant under a paired randomization test with Holm's correction. Code, results, per-query diagnostics and this paper's source are at `https://github.com/TheBenMeadows/dead-retriever`.

1 Introduction

A hybrid search system runs two retrievers over the same documents, typically a keyword index scored by BM25 and a dense embedding index searched by vector similarity, and fuses the two ranked lists into one. Reciprocal rank fusion [RRF; Cormack et al., 2009] is the usual way to fuse

them. With ranks starting at zero, it scores each document by the sum, over the retrievers that returned it, of $1/(k+r)$, where r is the document's rank in that retriever's list and k is a constant, 60 by convention.

Fusion is forgiving by design. If one retriever returns nothing, the fused list is simply the other retriever's list, and the user still sees a full page of results. That is the right behaviour for a query one retriever cannot answer. It is also the reason a broken retriever can go unnoticed: nothing downstream of the fusion distinguishes "the keyword retriever found nothing relevant" from "the keyword retriever is broken".

The fused scores show something narrower than health: at $K \leq k$, whether the two lists shared any document at all. A document returned by only one retriever scores at most $1/k$ (rank 0 gives exactly $1/k$). A document returned by two retrievers, each cut at depth K, scores at least $2/(k + K - 1)$, and that is above $1/k$ exactly when $K \leq k$. So, at $K \leq k$, a fused list whose maximum score is $1/k$ contains no document that both retrievers returned. It does not say that either list is empty. The per-retriever *row counts*, the number of documents each retriever returned for the query, separate an empty list from a short one and from two full lists that happen to share nothing. The row counts cost a counter in each retriever; the maximum fused score costs nothing, because the fusion has already computed it, and it can be read from a log that kept only scores. A system that fuses should record both.

I came to this through an incident. A personal search engine of mine fused an SQLite FTS5 keyword index with a dense index at $K = k = 60$. Its keyword query builder quoted every token of the query, and FTS5 reads a sequence of quoted phrases as an AND over all of them, so a conversational query of six or seven words almost never matched a 1 400-character chunk. The dense retriever filled every result list, and short technical queries, the kind used in spot checks, still matched. The row counts were first logged on 2026-08-18: zero on every conversational query. The maximum fused score on those queries had been 1/60 all along, and eleven days earlier the scores had been written down as "flat near the top".

The contributions are:

1. the derivation and its limits (Section 2): the equivalence needs $K \leq k$; one direction holds at any depth; at $K > k$ a document shared at deep ranks can hide below the ceiling; with two retrievers an empty list can never be missed while the other list is nonempty;
2. the incident (Section 3): the bug, the fix, and before-and-after counts from the production system;
3. a measurement on six BEIR collections with three encoders, 22 362 hybrid query runs (Sections 4 and 5): how often the buggy builder returns nothing, whether the ceiling catches it, where the ceiling and the row counts disagree and why, one collection on which the test is blind by construction, and whether repairing the keyword retriever improves retrieval;
4. what a system that fuses should log (Section 6).

Section 7 places the ceiling among the fusion and monitoring literature, and Section 8 states the limits.

2 The ceiling

Let L retrievers each return a ranked list cut at depth K, with every document appearing at most once per list. Let k be a positive integer, and let $r_\ell(d) \in \{0, \ldots, K-1\}$ be document d's 0-based rank in retriever ℓ's list when d is in it. Reciprocal rank fusion scores

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

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

$$\frac{2}{k+K-1} > \frac{1}{k} \iff K \le k.$$

So with $K \le k$, every document found by two or more retrievers outscores every document found by one, and for any fused list with at least one result:

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

I call a list with $\max_d s(d) \le 1/k$ a list *at the ceiling*. A nonempty fused list always contains some retriever's rank-0 document, which scores at least $1/k$, so a nonempty list at the ceiling has maximum exactly $1/k$. Throughout, the scores are the unweighted RRF sums before any later filtering or rescaling, and the test is applied as $\max_d s(d) \le 1/k + 10^{-12}$. 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 $K \le k+1$.

Three qualifications bound what a ceiling means in practice.

It is a statement about overlap, not about either retriever. Two retrievers that both return K documents and share none of them produce the same ceiling as one retriever and one empty list. So does a keyword list of one to three documents that the dense retriever did not return. In the incident (Section 3), the keyword retriever returned rows on four of the ten replayed queries and the fused list was at the ceiling on all ten: those four keyword lists held one to three chunks, none of them in the dense top 60. The ceiling says no document was reinforced by agreement. The row counts say whether each list was empty, short or full; at $K \le k$, the ceiling says the lists shared no document, whatever their lengths.

The equivalence needs $K \le k$. A document at 0-based ranks r_1 and r_2 in two lists scores above $1/k$ if and only if $r_1 r_2 < k^2$. Ranks 60 and 60 score exactly 1/60; ranks 90 and 90 score 2/150. A maximum of $1/k$ therefore rules out a shared document at rank 0 of either list at any depth, because that document's rank product is 0. At $K \le k$ it rules out every shared document, because $r_1 r_2 \le (K-1)^2 < k^2$. At $K > k$ it leaves open any shared document whose ranks multiply to at least k^2. One direction survives at any depth: a maximum above $1/k$ proves that some document was returned by two retrievers. A system that keeps per-retriever ranks should count shared documents directly. The experiments in Section 5 run at $K = 100 > k = 60$, report both, and measure how often the two disagree.

With two retrievers, an empty list cannot be missed. If one of two lists is empty and the other is not, every fused document is a singleton and the list is at the ceiling. So with two retrievers, as long as the other retriever returned something, the test has no false negatives for an empty list; everything it gets wrong is in the other direction, a ceiling over a nonempty list. With three or more retrievers an empty list can hide behind two nonempty lists that overlap. The equivalence itself holds for any L.

3 The incident

3.1 The system and the bug

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 RRF at $k = 60$.

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 conversational query of six or seven tokens almost never matched. Two details made things worse than "every token present" suggests. The builder's tokeniser kept hyphens, dots and some punctuation inside a token, while FTS5's own tokeniser splits on them, so a token such as `il-6` became a two-word phrase and therefore an adjacency constraint. And the code caught and ignored FTS5 syntax errors, so a query the grammar rejected would also have returned nothing, without a trace. Queries of two or three technical tokens did match, which is why spot checks during development passed, and the dense retriever returns K documents whenever the corpus has that many, so result lists were always full.

3.2 How it was found, and what the scores had already said

On 2026-08-18, while looking into why the keyword side rarely seemed to matter, I logged the keyword retriever's row counts for ten real queries. All five conversational queries matched zero chunks, and so did a two-word place-name query. Only queries of two or three technical tokens matched anything. The builder had been the same line of code since the system was built, and all four sibling engines carried it. That it had behaved this way on prose queries throughout is an inference from the unchanged code and the replay, since the system keeps no query log.

The fused scores had shown zero overlap on every replayed query. Over the fused lists of those ten queries (at most 120 documents each at $K = 60$, fewer after result assembly dropped the lower-ranked chunks of a document already listed, which cannot lower the maximum), the maximum score was exactly $1/60 = 0.01\overline{6}$. The values ran $1/60, 1/61, 1/62, \ldots$, with gaps where a chunk had

been dropped: every fused score was a single term $1/(k+r)$, so no document had been credited by both retrievers. Eleven days earlier, a note written while testing a recency-decay feature had recorded that the RRF scores were "flat near the top", adjacent ranks differing by about 1.4%, and had read that as a property of fusion. Flatness is a property of RRF; it was the wrong thing to look at. Consecutive singleton scores fall by a relative step of $1/(k+r+1)$: 1.6% at rank 0 and 1.4% near rank 10. Two identical nonempty lists would show the same steps at twice the score. What marked the ten queries was the scale, not the step: nothing scored above 1/60.

The fix drops stopwords and tokens shorter than three characters, joins the rest with `OR` and ranks by BM25. The drop is for cost: a raw OR over all tokens matched about 100 000 chunks in 161 ms per query; after it a typical query matches about 3 500 chunks in 2 to 4 ms. If no token survives the drop, the raw tokens are used. Section 5 measures both variants on public data. Table 1 gives the before-and-after counts. The corpus is private, so these aggregates (`results/production-aggregates.json` in the released repository) are the only production figures in the paper; Section 5 repeats the measurement on public collections. The same builder sat 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 retriever returned any row	4 / 10	10 / 10
Queries with a document found by both retrievers	0 / 10	10 / 10
Maximum fused score over the fused list	$1/60 = 0.0167$	0.025 – 0.033
Keyword retriever latency, conversational queries	–	2 – 4 ms

Table 1: The keyword retriever 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 keyword retriever returned rows, it returned one to three chunks, none of them in the dense top 60.

4 Experimental setup

To check that the effect is a property of the query builder and not of one private corpus, six retrieval conditions were run on six BEIR collections [Thakur et al., 2021] with three encoders.

Collections. SciFact [Wadden et al., 2020] (claim sentences against scientific abstracts), NFCorpus [Boteva et al., 2016] (short nutrition titles against medical documents), ArguAna [Wachsmuth et al., 2018] (whole arguments against counter-arguments), SciDocs [Cohan et al., 2020] (paper titles against abstracts), FiQA [Maia et al., 2018] (finance questions against forum posts) and TREC-COVID [Voorhees et al., 2020] (COVID-19 questions against CORD-19 abstracts). Table 2 gives their sizes. Data came through `ir_datasets` [MacAvaney et al., 2021]; only queries with relevance judgements are run.

Encoders. `nomic-embed-text-v1.5` [Nussbaum et al., 2024], the public release of the production encoder; `bge-base-en-v1.5` [Xiao et al., 2023]; and `e5-base-v2` [Wang et al., 2022]. All three are 768-dimensional, and each is used with the query and document prefixes its model card prescribes (Appendix B). Documents are embedded as title and text joined by a space; vectors are normalised and searched by exact cosine.

Keyword retriever. The builder's tokeniser is the regular expression `[\w$#.@-]+` over the lowercased query. All three builders quote each token. The buggy builder joins the quoted tokens with spaces (implicit AND). The fixed builder drops the 118 stopwords listed in Appendix B and every token under three characters, then joins the rest with `OR`, falling back to the raw tokens when nothing survives; I call the tokens it keeps the *content tokens*. A third builder joins every raw token with `OR`. The FTS5 index holds title and text columns with default BM25 weights.

Conditions. BM25-AND (the buggy builder), BM25-OR-RAW (OR over every token), BM25-OR (the fix), DENSE, HYBRID-AND (dense + buggy keyword, fused by RRF) and HYBRID-OR (dense + fixed keyword, fused by RRF). Depth $K = 100$ for every retriever and $k = 60$ for fusion, so the ceiling is read in its one-sided form and overlap is also counted directly from the ranks. The 18 collection–encoder pairs give $3\,727 \times 3 \times 2 = 22\,362$ hybrid query runs. The three encoders rerun the same queries, so they are not independent replications.

Diagnostics. For each hybrid condition the runner records, per query: whether the keyword list was empty, how many documents it held, whether the fused list was at the ceiling, the maximum fused score itself, and how many documents both retrievers returned. The runner computes them from the two retriever lists and the fused list.

Effectiveness. nDCG@10 and Recall@100 through `pytrec_eval` [Van Gysel and de Rijke, 2018]. Before scoring, a document whose id equals the query's id is removed, which is the default of BEIR's evaluator; this touches 1 298 of ArguAna's 1 406 queries and 55 of FiQA's 648, and no query elsewhere, and FiQA's figures are identical with and without the rule. The 108 ArguAna queries whose text is a corpus document under another id are not touched. Differences between conditions are tested per collection–encoder pair with a two-sided paired randomization test over queries (sign flips, 10^5 trials; Smucker et al., 2007) and Holm's correction within each family of 18 pairs (six families: two metrics by three comparisons; no correction across families). A difference is called significant at $\alpha = 0.05$ after correction. Per-query scores and the test's output are in `results/`. Differences quoted in the text are differences of the printed three-decimal values unless a fourth decimal is given. Every run was made in one Kaggle kernel on an NVIDIA T4 (Appendix B).

5 Results

Tables 5 and 6 give dense and hybrid effectiveness; Appendix A supplies the keyword-only results, the keyword query times and the shared-document counts.

5.1 How often the buggy builder returns nothing

Outside ArguAna, which Section 5.3 treats separately, the buggy builder's empty rate runs from 50.8% on NFCorpus to 98.0% on SciFact (Table 2). Mean query length does not order the five rates: NFCorpus has the shortest queries and the lowest rate, but TREC-COVID's questions are about as long as FiQA's and SciDocs's and its rate is lower. Content tokens per query order all five exactly. Every quoted token is one more condition a single document must meet; stopwords are met almost everywhere; the content tokens are the conditions that fail. By raw token count SciDocs is fourth of the five; by content tokens it is second, where its empty rate sits.

			tokens per query		quoted AND		OR after the drop	
Collection	docs	queries	all	content	kw empty	at ceiling	kw empty	at ceiling
SciFact	5 183	300	12.6	8.7	98.0%	98.0%	0.0%	0.0%
NFCorpus	3 633	323	3.3	2.5	50.8%	51.4–51.7%	7.7%	7.7–8.7%
ArguAna	8 674	1 406	195.1	106.1	0.0%	0.0%	0.0%	0.0%
SciDocs	25 657	1 000	9.4	7.3	94.2%	94.7–94.9%	0.0%	0.1%
FiQA	57 638	648	11.1	6.2	81.8%	87.2–88.4%	0.0%	0.0–0.3%
TREC-COVID	171 332	50	10.6	5.7	74.0%	88.0–90.0%	0.0%	0.0%

Table 2: The keyword list and the ceiling on six BEIR collections. "tokens per query" is the mean query length in the builder's own tokens, all of them and the content tokens the fixed builder keeps. "kw empty" is the fraction of queries on which the keyword retriever returned no document; it does not depend on the encoder. "at ceiling" is the fraction on which the fused list was at the ceiling, given as the range over the three encoders where they differ.

BM25 with the fixed builder is a usable retriever on every collection (Appendix A). After the fix, only NFCorpus has queries that match nothing: 25 title queries (7.7%) in which no token matches any document, so the raw OR is empty on the same 25. The stopword drop that made the fix cheap in production saves time on every collection and costs little effectiveness. BM25-OR-RAW is 2.2 to 18 times slower than BM25-OR (Table 7); the extreme is ArguAna, 905 against 52 ms on queries that average 195 tokens. It is worse at nDCG@10 by 0.001 to 0.011 on five collections. The exception is FiQA, where OR over every token is better by 0.009 nDCG@10 and 0.013 Recall@100, at 5.6 times the cost.

5.2 Does the ceiling catch an empty keyword list?

With two retrievers it must, whenever the other list is nonempty. An empty keyword list then puts the fused maximum at exactly $1/k$, so the set of empty-list queries is contained in the set of at-ceiling queries, and across all 22 362 hybrid query runs the ceiling missed no empty list. On the 294 of SciFact's 300 queries with an empty keyword list, the per-query rows show that the HYBRID-AND list is the DENSE list: the same document count and a maximum score of exactly 1/60.

Everything the ceiling and the row count disagree on is therefore a ceiling over a nonempty keyword list. There were 176 such query runs, 164 under the buggy builder and 12 under the fixed one, and none on SciFact or ArguAna. Table 3 gives the buggy builder's counts by collection, as a share of the queries on which its keyword list was nonempty.

In 174 of the 176 the two retrievers shared no document, so the ceiling was right as a zero-overlap test and wrong as an empty-list alarm. The other two are the case Section 2 said $K > k$ leaves open: on one FiQA query for bge and one for e5, under the fixed builder, both retrievers returned 100 documents and shared exactly one, ranked deep enough in both that its fused score did not exceed $1/k$. Two such runs in 22 362 are why a system that runs at $K > k$ should count shared documents directly.

The at-ceiling rate among nonempty keyword lists was higher for shorter lists (Figure 1). Pooled over every collection except ArguAna, where the keyword list is never disjoint, and over both builders, the fused list was at the ceiling on 137 of 726 runs with a keyword list of one to three documents (18.9%), on 35 of 588 with four to 49 (6.0%), and on 4 of 6 636 with 50 or more (0.06%;

	buggy builder, per encoder			lists of 1–3 documents	
Collection	nonempty lists	at the ceiling	share	runs	at the ceiling
SciFact	6	0	0.0%	18	0
NFCorpus	159	2–3	1.3–1.9%	330	13
ArguAna	1 406	0	0.0%	4 218	0
SciDocs	58	5–7	8.6–12.1%	141	18
FiQA	118	35–43	29.7–36.4%	219	90
TREC-COVID	13	7–8	53.8–61.5%	18	16

Table 3: Ceilings over a nonempty keyword list. Left: queries on which the buggy builder's keyword list was nonempty (the count does not depend on the encoder), how many of them were at the ceiling, and the share, as ranges over the three encoders. Right: hybrid query runs in which the keyword list held one to three documents, pooled over both builders and all three encoders, and how many of those runs were at the ceiling.

two disjoint lists and the two FiQA runs above). The buggy builder supplies most of the short lists: 612 of the 726 runs in the first bucket, at the ceiling on 132 of them, against 5 of 114 for the fixed builder. Among lists of one to three documents, the at-ceiling share is highest on SciDocs, FiQA and TREC-COVID, the three largest collections (Table 3, right). This ordering does not establish an effect of corpus size: SciFact's 0 of 18 is consistent with NFCorpus's rate, and the collections differ in more than corpus size. On the three largest collections, keyword lists of one to three documents were disjoint from the dense top 100 on 12.8, 41.1 and 88.9% of runs (124 of 378 pooled). The ceiling also appeared over a keyword list of four or more documents on 39 runs, all on FiQA, TREC-COVID and SciDocs, so "a short list" is not a safe reading of every such ceiling; the row count is.

5.3 A keyword list that is never empty and almost never relevant

ArguAna's queries are whole arguments, 195 tokens on average, and the quoted-AND builder never returned nothing for them: it returned one document for 1 402 of the 1 406 queries and two for the other four. The reason is the collection's construction. Every query's text is also a document in the corpus, and a query is the one text guaranteed to contain all of its own tokens, so the AND builder finds the query itself; I call that document the query's *self-match*. On every query the AND list held a document whose text equals the query's after whitespace and case normalisation: under the query's own identifier on 1 298 (at the top of the list on 1 296) and under another identifier on the other 108. A query's own document is not one of its relevant documents, and the AND list held a relevant document for one query in 1 406.

The ceiling cannot see this. The dense retriever also returns that document in its top 100 on every query for every encoder, so the two retrievers share it, the fused maximum is above $1/k$ on every query, and the ceiling never appears: Table 2 shows 0.0% for every encoder. Fusion with this keyword list changed nothing measurable: HYBRID-AND and DENSE agree to every decimal the result files carry in both metrics for all three encoders. The ceiling tests for agreement, and a retriever that agrees with the other on one useless document passes it. Per-retriever row counts do not catch it either, since the list is nonempty on every query. What catches it is the distribution of row counts, one document on 99.7% of queries, which is as much an alarm as zero. ArguAna's construction is unusual, but a corpus that indexes the kind of text it is queried with is not: a search engine over a mailbox, a notes vault or a ticket tracker can receive the text of an indexed item as a

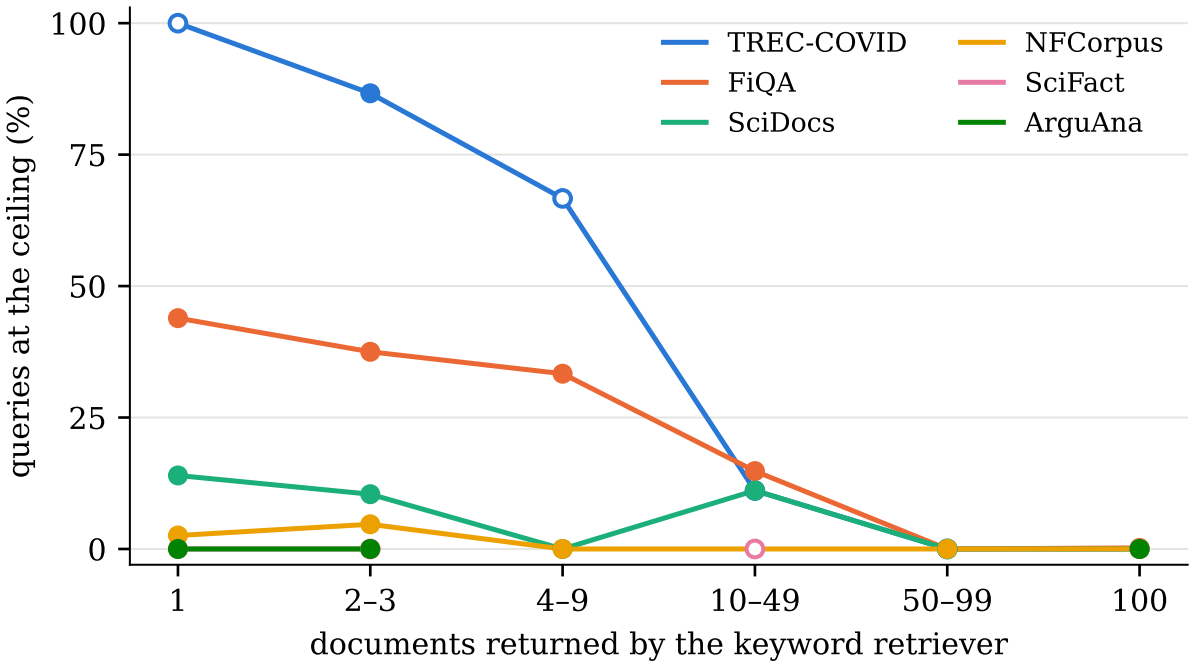


Figure 1: Fraction of hybrid query runs at the ceiling, by the number of documents the keyword retriever returned, pooled over both builders and all three encoders. A hollow marker is a cell with fewer than ten runs; a gap is a cell with none. SciFact and ArguAna sit at zero in every populated cell, under the other lines. Empty keyword lists, at 100% by construction, are not drawn.

query. Whether the self-match is then a failure depends on the task; when finding the source item is the point, it is the answer.

5.4 Short lists under control

To measure the ceiling against short nonempty lists under control, 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 SciFact and NFCorpus query where the keyword retriever returned anything, for bge and nomic (Table 4). A cap is not a length: a query whose keyword list held fewer than n documents keeps its short list in every cell, so each cell is the ceiling rate under a cap of n over the same 300 and 298 queries.

In every cell, for both encoders, the ceiling appeared exactly on the queries where the two lists shared no document. At $K = 60$ a cap of one document leaves the lists disjoint on 8.7 to 15.1% of these queries and a cap of twenty or more on at most 1.7%; raising the dense depth to 100 never raises a figure. 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, 2 392 query runs, it did not occur. It did occur twice in the natural FiQA runs above, both with $n = 100$, so the sweep's zero is a property of these two small collections, not of the test. The sweep was not repeated on the four larger collections; the natural row counts in Section 5.2 stand in for it there, and they show the short-list rate is several times higher on three of them than these two suggest.

	SciFact		NFCorpus	
n	$K = 60$	$K = 100$	$K = 60$	$K = 100$
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 4: Fraction of queries with a nonempty keyword list on which the fused list was at the ceiling 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 queries). In every cell this fraction equals the fraction of queries whose two lists were disjoint, and the fraction at the ceiling 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`.

5.5 Does repairing the keyword retriever improve retrieval?

	bge-base-en-v1.5			nomic-embed-text-v1.5			e5-base-v2		
Collection	dense	h-and	h-or	dense	h-and	h-or	dense	h-and	h-or
SciFact	0.740	0.741	0.723	0.703	0.705	0.708	0.719	0.722	0.726
NFCorpus	0.374	0.382	0.363	0.347	0.361*	0.344	0.355	0.368*	0.352
ArguAna	0.641	0.641	0.606*	0.483	0.483	0.517*	0.447	0.447	0.532*
SciDocs	0.217	0.216	0.195*	0.187	0.187	0.182	0.187	0.186	0.183
FiQA	0.406	0.396*	0.348*	0.376	0.367*	0.330*	0.399	0.389*	0.362*
TREC-COVID	0.779	0.744*	0.763	0.823	0.784	0.769	0.696	0.669	0.714

Table 5: nDCG@10 of the dense retriever, the hybrid with the buggy keyword retriever (h-and) and the hybrid with the fixed one (h-or). A star marks a hybrid whose paired difference from the dense retriever of the same encoder is significant at $\alpha = 0.05$ after Holm's correction (two-sided randomization test, 10^5 sign flips, 18 pairs per family).

Repairing the keyword retriever raises overlap: the mean number of documents both retrievers returned rose from at most 4.0 per query (Table 8) to between 19.0 and 43.7 in every collection–encoder pair. It does not reliably improve retrieval (Tables 5 and 6). Over the 18 pairs, the repaired hybrid is above DENSE at nDCG@10 in 5 and below it in 13, and the paired test calls 2 of the gains and 5 of the losses significant. At Recall@100 it is above in 9 and below in 9, with one gain and three losses significant; one of the nine gains, NFCorpus with bge, is 0.0006, inside the table's rounding. The buggy hybrid is above the repaired one at nDCG@10 in 12 (six significant) and below it in 6 (two significant).

The two significant gains over DENSE are on ArguAna, where BM25 alone reaches 0.484 nDCG@10, level with nomic (0.483) and above e5 (0.447): the repaired hybrid gains 0.085 for e5 and 0.034 for nomic. It loses 0.035 for bge, also significantly; bge's dense retriever (0.641) is far above BM25. The other three gains are small and not significant (0.005 and 0.007 on SciFact for nomic and e5; 0.018 on TREC-COVID for e5). The largest single loss, and the only collection where every encoder loses significantly, is FiQA, where BM25 is far behind every encoder

	bge-base-en-v1.5			nomic-embed-text-v1.5			e5-base-v2		
Collection	dense	h-and	h-or	dense	h-and	h-or	dense	h-and	h-or
SciFact	0.967	0.967	0.969	0.928	0.928	0.958	0.963	0.963	0.959
NFCorpus	0.337	0.341*	0.337	0.298	0.303*	0.309	0.319	0.323*	0.328
ArguAna	0.993	0.993	0.991	0.980	0.980	0.984	0.949	0.949	0.981*
SciDocs	0.496	0.495	0.469*	0.404	0.404	0.409	0.421	0.421	0.425
FiQA	0.741	0.740	0.715*	0.682	0.682	0.675	0.732	0.732	0.711
TREC-COVID	0.141	0.140	0.133	0.150	0.148	0.137*	0.128	0.127	0.124

Table 6: Recall@100, same conditions, layout and star as Table 5.

(0.224 nDCG@10 against 0.376 to 0.406) and the repaired hybrid is 0.037 to 0.058 below DENSE; TREC-COVID with nomic loses 0.054, not significant on 50 queries.

The buggy hybrid's 12 wins over the repaired one are mostly abstention. When the keyword list is empty, the hybrid is the dense retriever, and the buggy builder's list was empty on most queries of most collections. On SciFact, HYBRID-AND and DENSE differ by at most 0.003 nDCG@10 and 0.000 Recall@100, all of it from the six queries with a nonempty keyword list; on SciDocs, with 942 empty lists, by at most 0.001 in each. In 8 of the 12 the buggy hybrid is not above DENSE. In seven, its keyword list was empty on 74 to 94% of queries, so the hybrid was the dense list on most queries. The eighth is ArguAna with bge, where the keyword list was the self-match on every query and a second document on four, and both metrics equal the dense retriever's to every decimal. In all eight, the repaired keyword list's OR results diluted the top 10, and that dilution is what the "win" records. Only on NFCorpus (all three encoders) and SciFact with bge is the buggy hybrid above DENSE as well, by 0.001 on SciFact and by 0.008 to 0.014 on NFCorpus (significant for nomic and e5), where the AND list was nonempty on half the queries and held four or more documents on 86 of those 159. On TREC-COVID, by contrast, the buggy hybrid is 0.027 to 0.039 below DENSE, significant for bge ($p = 0.043$), on a difference carried by the 13 queries with a nonempty list.

Unweighted RRF treats a weak keyword list and a strong dense list alike, which is one reason Bruch et al. [2024] find a tuned convex combination competitive. Wang et al. [2021] on MS MARCO, by score interpolation, and Chen et al. [2022] on BEIR report gains from combining BM25 with dense retrieval. Whether RRF over BM25 and a given dense encoder helps is a question these runs answer only per pair. Overlap and effectiveness are different questions, and only relevance judgements answer the second.

6 What to log

Log, per query, each retriever's row count and the maximum fused score. With per-retriever depth $K \leq k$, a maximum at $1/k$ (or $1/(k+1)$ for one-based ranks) means the retrievers did not agree on any document for that query. At greater depth it means no document is shared at ranks r_1, r_2 with $r_1 r_2 < k^2$, and shared documents should be counted directly. Either way, read the row counts to learn whether a retriever's list was empty, short or full. A ceiling over a short nonempty list was common on the larger collections here, so a ceiling is a reason to read the row count, not a verdict.

Look at the distribution of row counts, not only at zero. A retriever that returns one document on nearly every query is as suspect as one that returns none, and if the corpus indexes the kind of text it is queried with (messages, notes, tickets), check whether that one document is the query itself and whether returning it is the task. 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 also return nothing. If the fusion scores look "flat", compute $1/(k + r + 1)$ before concluding anything about fusion. None of these signals is a verdict on effectiveness: on these collections a repaired keyword retriever made the hybrid significantly worse than the dense retriever alone at nDCG@10 in five pairs and significantly better in two, and that question needs relevance judgements.

7 Related work

Cormack et al. [2009] proposed reciprocal rank fusion and set $k = 60$ by experiment; the ceiling is arithmetic on their formula, and the system in the incident used their constant unchanged, which is what made a 1/60 maximum recognisable on sight. The older fusion literature studied the overlap between the lists being fused. CombSUM sums the scores a document receives and CombMNZ multiplies that sum by the number of systems that returned the document [Fox and Shaw, 1994], so CombMNZ rewards agreement explicitly. Lee [1997] showed that fusion helps when the systems agree on relevant documents more than on non-relevant ones; Aslam and Montague [2001] modelled metasearch combination, and Montague and Aslam [2001] treated the score-scale problem that rank fusion sidesteps. Nuray and Can [2006] use data fusion to rank retrieval systems without relevance judgements, by each system's agreement with the fused output; the ceiling asks only whether the input lists share any document. Rank-biased overlap [Webber et al., 2010] is the principled measure of agreement between two indefinite rankings; the count of shared documents in Section 5 is its crudest relative. On hybrid lexical and dense retrieval, Luan et al. [2021] and Gao et al. [2021] establish the complementarity a hybrid relies on, Wang et al. [2021] measure what interpolating BM25 adds to BERT-based dense retrievers on MS MARCO passage ranking, Chen et al. [2022] what a BM25 and dense hybrid gains zero-shot on BEIR, and Bruch et al. [2024] analyse fusion functions and show when RRF is and is not competitive with a tuned convex combination. The ceiling applies to unweighted RRF scores as assigned; it does not bear on which fusion function to choose.

That keyword retrieval does badly on long, verbose queries is an old observation: Kumaran and Allan [2007] shorten such queries and Bendersky and Croft [2008] pick out their key concepts, and the stopword drop of Section 4 is a crude fixed-rule version of that reduction. 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 logging 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. I know of no earlier statement of the $1/k$ maximum as an overlap test, and of no published report of a silently inert retriever inside a hybrid; that is what I could find, not evidence that there is none.

8 Limitations

The production measurements used here are on ten queries over a private corpus and are reported as aggregates the reader cannot check. They establish that the failure happened and how it looked, not an effect size, and the claim that the keyword retriever had been inert since the system's first version is an inference from unchanged code.

The public reproduction uses six collections, three base-sized 768-dimensional encoders, and FTS5's default BM25 weighting of title and text. The three encoders rerun the same queries, so the 18 pairs are not 18 independent samples, and the paired tests are corrected within each family of 18 but not across the six families; corrected over all 108 tests at once, 23 of the 31 differences significant within their families remain so, among them all seven significant comparisons of the repaired hybrid with the dense retriever at nDCG@10. The collections differ in domain, corpus size, document length, relevance style and query length at once, so the orderings reported here (empty rate against content tokens per query, short-list disjointness across collections of different size) are observations over five or six points, not controlled effects, and the pooled row-count rates mix the two builders, whose short lists arise differently. The truncation sweep covers only the two smallest collections.

The exact ceiling statement needs $K \leq k$, and the guarantee that an empty list is never missed needs two retrievers and a nonempty counterpart; the common $K = 100$, $k = 60$ setting gets the one-sided direction and the direct overlap count instead, and it produced two ceilings with overlap present. The ceiling detects zero overlap; attributing it to an empty list needs the per-retriever row counts, and a retriever that returns one useless document the other retriever also returns at a high rank, as on ArguAna, passes both the ceiling and a zero-row check.

9 Conclusion

With per-retriever depth $K \leq k$, a maximum reciprocal-rank-fusion score of $1/k$ over a result list means no document in it was returned by more than one retriever, and a system that fuses can read that from the scores it already computes. In the incident, the ceiling held on all ten replayed queries; eleven days earlier the scores had been noted only as flat. On six BEIR collections the ceiling missed no empty keyword list in 22 362 runs, as it must with two retrievers and a dense list that was never empty, and where it disagreed with the row counts it was reporting a nonempty list that shared nothing with the dense list, except twice, at $K > k$, where a shared document sat too deep to show. It cannot see a retriever whose one returned document the other retriever also returns, as on ArguAna; the distribution of row counts can. Repairing the keyword retriever raised overlap in every pair and improved nDCG@10 over the dense retriever alone in 5 of 18, two of them significantly, and lowered it in 13, five significantly. At $K \leq k$ the scores can say whether two retrievers agreed on anything, and at greater depth a direct count of shared documents can; the row counts say what that means; whether fusing a retriever helps is a question for relevance judgements.

Use of AI tools

The code, the experiment and audit scripts, and the text of this paper were drafted with Claude (Anthropic) in Claude Code under the author's direction. Drafts were reviewed by other language models through the Devin, Cursor and Kagi command-line tools; every defect a reviewer reported was reproduced or refuted by a script before it changed the paper, and the review records, the probes and the audit script that checks every number in the text against the result files are in the repository. The author operated the production system, chose what to measure, ruled on every claim and takes responsibility for all content. This work received no funding, and the author declares no competing interests.

References

Javed A. Aslam and Mark Montague. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 276–284, 2001.

Michael Bendersky and W. Bruce Croft. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–498, 2008.

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.

Tao Chen, Mingyang Zhang, Jing Lu, Michael Bendersky, and Marc Najork. Out-of-domain semantics to the rescue! Zero-shot hybrid retrieval models. In *Advances in Information Retrieval: 44th European Conference on IR Research (ECIR 2022)*, pages 95–110, 2022.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. SPECTER: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, 2020.

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.

Giridhar Kumaran and James Allan. A case for shorter queries, and helping users create them. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 220–227, 2007.

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.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. WWW'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of The Web Conference 2018*, pages 1941–1942, 2018.

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.

Rabia Nuray and Fazli Can. Automatic ranking of information retrieval systems using data fusion. *Information Processing & Management*, 42(3):595–614, 2006.

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.

Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pages 623–632, 2007.

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.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. TREC-COVID: Constructing a pandemic information retrieval test collection. *ACM SIGIR Forum*, 54(1):1–12, 2020.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251, 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.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Shuai Wang, Shengyao Zhuang, and Guido Zuccon. BERT-based dense retrievers require interpolation with BM25 for effective passage retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 317–324, 2021.

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 All conditions, all collections

Tables 7 and 8 complete Tables 2, 5 and 6: the three BM25-only conditions, which do not depend on the encoder, and the mean number of documents both retrievers returned under each hybrid. Query times are mean wall-clock per query for the keyword search on the Kaggle host's shared CPU, with query encoding excluded, from the `bge-base-en-v1.5` run of each collection; they compare conditions within a collection and only loosely across collections. Every figure is in `results/*.json`.

B Run details

All 18 collection–encoder pairs were run in a Kaggle kernel on one NVIDIA T4 (`kaggle/run.py` in the repository, which clones the runner at a pinned commit and records package versions and the device in a manifest): PyTorch 2.11 with CUDA 12.8, `transformers` 4.49, `sentence-transformers` 3.4.1, `ir_datasets` 0.6.3. The four SciFact and NFCorpus pairs with bge and nomic were first run on CPU, and the GPU runs reproduced them to every digit the result files carry for nDCG@10 and Recall@100; every result file in the repository is from the Kaggle runs. Documents were embedded in batches of 64, except with nomic, which accepts 8 192 tokens where the other two encoders stop at 512 and ran out of GPU memory at that size on the three largest collections, so every nomic run used batches of 8; between batch 64 and batch 8 the nomic runs on SciFact were identical and on NFCorpus one query's fused maximum score changed in the fourth decimal, with no printed figure affected. `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. The encoder's document prefix is applied where its model card prescribes one (`search_document:` for nomic, `passage:` for e5, none for bge); queries carry the query prefix (`search_query:` for nomic; `query:` for e5; `Represent this sentence for searching relevant passages:` for bge, which its card

Collection	Condition	nDCG@10	R@100	kw empty	ms/query
SciFact	bm25-and	0.018	0.018	98.0%	0.5
SciFact	bm25-or-raw	0.659	0.879	0.0%	13.6
SciFact	bm25-or	0.665	0.876	0.0%	4.1
NFCorpus	bm25-and	0.193	0.095	50.8%	0.2
NFCorpus	bm25-or-raw	0.304	0.232	7.7%	2.6
NFCorpus	bm25-or	0.306	0.233	7.7%	1.2
ArguAna	bm25-and	0.001	0.001	0.0%	34.5
ArguAna	bm25-or-raw	0.473	0.957	0.0%	905.4
ArguAna	bm25-or	0.484	0.965	0.0%	51.5
SciDocs	bm25-and	0.003	0.002	94.2%	1.0
SciDocs	bm25-or-raw	0.152	0.343	0.0%	41.9
SciDocs	bm25-or	0.153	0.344	0.0%	12.6
FiQA	bm25-and	0.014	0.015	81.8%	3.8
FiQA	bm25-or-raw	0.233	0.521	0.0%	103.6
FiQA	bm25-or	0.224	0.508	0.0%	18.6
TREC-COVID	bm25-and	0.075	0.003	74.0%	15.6
TREC-COVID	bm25-or-raw	0.532	0.091	0.0%	352.6
TREC-COVID	bm25-or	0.538	0.101	0.0%	110.2

Table 7: The three keyword-only conditions. "kw empty" is the fraction of queries on which the builder matched no document.

asks for on retrieval queries and on nothing else). The FTS5 index is built atomically. ArguAna, SciDocs and TREC-COVID have a single test split in BEIR and are loaded as `beir/arguana`, `beir/scidocs` and `beir/trec-covid`.

The fixed builder's stoplist, 118 words: *a an the and or but if then than that this these those of in on at to for from by with about into over under after before is are was were be been being am do does did doing have has had having i me my mine we us our you your he she it its they them their as not no nor so such can could should would will shall may might must what which who whom whose when where why how all any both each few more most other some only own same too very just dont don t s re ve ll m d get got make made want need know think see look.*

Per-query diagnostics are in `results/*.perquery.jsonl` and per-query effectiveness scores in `results/*.pereval.jsonl`; `scripts/keyword_facts.py` computes the encoder-free keyword facts, `scripts/tables.py`, `scripts/natural_lengths.py` and `scripts/figure.py` the tables, per-query counts and Figure 1, `scripts/significance.py` the paired tests, and the audit script `review/orrery-audit.py` checks every number in this paper against the result files. The two negative results measured on the production system beside the broken retriever (a recency-decay multiplier and a title prefix before embedding, both rejected on seven to fifteen queries) are recorded in `results/production-aggregates.json` and are not evidence about the ceiling.

	hybrid-and			hybrid-or		
Collection	bge	nomic	e5	bge	nomic	e5
SciFact	0.02	0.02	0.02	28.9	25.9	27.9
NFCorpus	4.0	3.7	3.8	21.3	21.0	20.6
ArguAna	1.0	1.0	1.0	38.6	43.7	33.5
SciDocs	0.13	0.17	0.15	29.3	32.3	31.7
FiQA	0.44	0.45	0.36	22.1	24.1	19.0
TREC-COVID	0.86	0.70	0.78	21.6	21.9	19.6

Table 8: Mean number of documents per query returned by both retrievers, for the hybrid with the buggy keyword retriever and the hybrid with the fixed one, by encoder.