

Season 2: Retrieval and RAG

Search.

Keyword, vector, or both. Each one is blind where the other sees. Five ways to search Harvard's library, tested against the 520 test questions.

One stage changes: the search

Everything else stays as it was after E4, including the new section chunks.

1 Load	2 Chunk	3 Embed	4 Rewrite	5 Search	6 Filter	7 Rerank	8 Prompt	9 Answer	10 Cite	11 Score
-----------	------------	------------	--------------	-------------	-------------	-------------	-------------	-------------	------------	-------------

After E4, search put the right passage in the top 5 for **281 of 400** questions. One kind of question still failed most of the time:

A REAL TEST QUESTION

What does error E-2194 mean? The answer is one sentence in one article

Error codes

Search found the right passage for only **15 of 40** questions about an error code or part number. For E-2194 it came 10th.

The question

Is there a better way to search, one that finds codes without losing what already works? And does it give more right answers?

What search does

1. Score

Give every chunk in the library a score for how well it matches the question. After E4 that is **644 chunks**.

2. Rank

Sort by score, best first, and pass the best **50** on. Later stages (filter, rerank) can reorder them.

3. Top k

Stage 8 puts the first **k** passages into the prompt. The model never sees the rest.

Is k configurable? Yes, one setting

A bigger k gives more chances the answer is included, but a longer, costlier prompt with more to distract the model. This episode keeps $k = 5$; E9 tests 3 and 8.

```
prompt: {k: 5, order: rank}
```

Our search number is "recall at k"

Of the 400 answerable questions, how many had the right passage in the top $k = 5$. This episode only changes step 1: **how a chunk is scored**.

Two ways to score a chunk

KEYWORD SEARCH

Count shared words

Split the question into words. A chunk scores for each word it contains. **Rare words count more**: "2194" appears in one article of the library, "error" in dozens.

what does error e 2194

mean

- + Exact codes, names and numbers
- Blind to a question in different words

VECTOR SEARCH (SINCE E2)

Compare meaning

Turn the question and every chunk into an embedding, a list of numbers for its meaning (S1 E3). A chunk scores by **how close its meaning is** to the question's.

[0.012, -0.031, 0.044, ...]

- + Questions worded differently from the document
- Codes that differ by a digit look almost the same

Next: how keyword search turns shared words into a score.
Then each method on a real question.

How keyword search scores: BM25

BM25 ("Best Matching 25") is a formula from the 1990s. It is still the standard keyword scorer, the default in search engines such as Elasticsearch and OpenSearch. Three ideas:

Rare words count more

A word in few chunks says a lot about them. A word in every chunk says nothing.

Repeats help, a little

Saying a word twice scores more than once, but each repeat adds less.

Long chunks scaled down

A long chunk does not win just by containing more words.

In one line

BM25 rewards a chunk for containing the question's **rarest** words. Next: the numbers, on a real question.

BM25, word by word

"What does error E-2194 mean?" split into words, each scored against the chunk that holds the answer.

WORD	CHUNKS CONTAINING IT, OF 644	POINTS
what, does	275, 308	0: everywhere
error	23	0: not in this chunk
mean	89	0: chunk says "means"
e	60	3.26
2194	1	8.72

Total 11.98: 1st by a clear margin

Almost all of it comes from one rare word. Note the catch: "mean" and "means" are different words to BM25. It matches spelling exactly, never meaning.

The runner-up, a Mira vacuum error article, scored 8.0, from "error" and "e". It has no "2194".

The code that meaning misses

QUESTION
What does error E-2194 mean? Answer: an interrupted factory reset on the Ashby tumble dryer

VECTOR SEARCH

1	Tessa dishwasher: saving energy	0.417
2	Tessa: warning light, E-4021	0.397
3	Tessa: delay start fault, E-9509	0.378
4	Tessa: not switching on, E-1736	0.373
5	Mira vacuum update, E-7804	0.372
...		
10	Ashby dryer reset, E-2194	0.356
15	Keel washer, E-4194	0.337
19	Orrin heat pump, E-1294	0.331

KEYWORD SEARCH

1	Ashby dryer reset, E-2194	12.0
2	Mira vacuum update, E-7804	8.0
3	Tessa dishwasher: saving energy	8.0
4	Vale fridge freezer warranty	6.1

1st, well clear

10th: not in the prompt

Why

To an embedding, every "what does error E-something mean" article means much the same thing, so the right one sits among them. Keyword search matches the rare word **2194**, which only one article contains. Across all 40 code questions: vector found **15**, keyword **40**.

The question in other words

QUESTION

Which wireless frequency does my smart water boiler support? Answer: 2.4 GHz

The passage that answers it

"The Brio kettle connects to a 2.4 GHz home wi-fi network."

"Smart water boiler" means kettle. "Wireless" means wi-fi. A customer asks in their words, not the manual's.

KEYWORD SEARCH

Not in the top 50

The question and the passage share **no words at all**, so the passage scores zero. Keyword search ranked other kettle articles that happen to say "wireless".

VECTOR SEARCH

1st

The meanings are close even though the words are not. This is what embeddings are for.

Across all 40 questions like this: vector found **40**, keyword **0**.

Most questions: both work

OUR QUESTION

How much longer does Extended Care keep my warranty going? Answer: 58 months

Our question shares its words **and** its meaning with the passage: "Extended Care", "warranty". Every search method puts the answer **2nd**, in the prompt.

Shared words	40 test questions like ours: vector found 40, keyword 39.
Codes	Keyword wins: 40 against 15.
Other words	Vector wins: 40 against 0.

So use both?

That is hybrid search: run both, then merge the two rankings into one. The question is how to merge them without inheriting both blind spots.

Stage 5: Search

WHAT CHANGES IN THIS STAGE

Hybrid search runs keyword and vector search, then merges the two rankings with **reciprocal rank fusion**. It uses only each chunk's **place** in each list, not its score, because the two kinds of score are on different scales (12.0 against 0.42).

```
E4 search: {method: vector, depth: 50}
```

```
E5 search: {method: hybrid, depth: 50, fuse_depth: 20}
```

src/lab/rag/search.py

the whole function

```
def fuse(rankings, k=60):
    """Reciprocal rank fusion: each ranking gives a document
    1 / (k + rank), and the scores add up."""
    scores = {}
    for ranking in rankings:
        for rank, doc in enumerate(ranking, start=1):
            scores[doc] = scores.get(doc, 0) + 1 / (k + rank)
    return sorted(scores, key=scores.get, reverse=True)
```

fuse_depth: 20

How far down each list fusion looks. Only each list's top 20 chunks take part. A chunk ranked 21st or lower in a list scores nothing from that list.

How fusion decides: agreement wins

Each list gives a chunk points for its place: $1 / (60 + \text{place})$. Look how little place matters:

1st	2nd	5th	10th	20th	not listed
0.0164	0.0161	0.0154	0.0143	0.0125	0

The lesson

1st place in one list is worth barely more than 20th. What really counts is being in **both** lists: that roughly doubles the score. Fusion rewards chunks the two methods agree on.

ON E-2194: THE ANSWER IS 10TH FOR VECTOR, 1ST FOR KEYWORD

CHUNK	VECTOR	KEYWORD	POINTS	FUSED
Tessa: saving energy	1st	3rd	$0.0164 + 0.0159 = 0.0323$	1st
Mira vacuum, E-7804	5th	2nd	$0.0154 + 0.0161 = 0.0315$	2nd
Ashby dryer, E-2194	10th	1st	$0.0143 + 0.0164 = 0.0307$	3rd
Tessa: E-4021	2nd	–	0.0161	4th

Both lists contain the answer, so it climbs from 10th to **3rd**: into the top 5, and the model now answers correctly.

The test: five searches

Every result from here to the end of the episode compares these five. Each one is a different stage 5; nothing else in the system changes.

Vector the reference Today's system, unchanged. Everything is compared against it.	
Keyword BM25 alone.	Hybrid, all Both, fusing all 644 places in each list.
Hybrid, top 50 Both, fusing each list's top 50.	Hybrid, top 20 Both, fusing each list's top 20.

What we want to learn
Does keyword search find the codes vector misses? Does vector find the paraphrases keyword misses? And can a hybrid keep both strengths, and turn them into more right answers?

Two rounds, one rule set in advance

1 Round 1: search only, all five

On all 400 answerable questions. Checked by code, so no model and no cost. Results on slides 14 and 15.

→ Pick one hybrid to run in full

Chosen on the 100 separate dev questions. All three tied, so the tie-break written in advance took the smallest: top 20.

2 Round 2: the full system

Hybrid top 20 against vector, end to end on all 520 test questions, every answer judged. Results on slides 16 to 18.

The rule to join, written before any results

The hybrid joins the system only if its **search number** is clearly above vector's (E3's check: the likely ranges must not overlap), and its right answers are not lower. Keep this in mind: it decides the ending.

Each method is blind where the other sees

Search: right passage in the top 5. Each column is one kind of test question, 40 of each; the last is all 400.

	CODES	OTHER WORDS	SHARED WORDS	FOLLOW-UPS	ALL 400
Vector the reference	15	40	40	7	281
Keyword	40	0	39	4	232
Hybrid, all	31	0	39	0	251
Hybrid, top 50	31	6	39	0	257
Hybrid, top 20	32	23	39	0	276

Read across	Keyword finds every code and no paraphrase. Vector is the reverse. Both handle shared words.
Read down	The hybrids keep most of keyword's codes, but lose paraphrases, and the more they fuse, the more they lose.

No method beat vector's 281 overall. Follow-ups such as "And the Tessa dishwasher?" fail everywhere: E6 rewrites them.

The fusion trap

Back to the kettle question. Vector put the answer 1st. Keyword never found it. Hybrid, top 20:

CHUNK	VECTOR PLACE	KEYWORD PLACE	SUM	HYBRID
Brio kettle: firmware	7	2	$1/67 + 1/62 = 0.0311$	1st
Brio kettle: voice pairing	6	4	$1/66 + 1/64 = 0.0308$	2nd
Brio kettle: not in app	4	7	$1/64 + 1/67 = 0.0306$	3rd
Brio kettle: wi-fi, the answer	1	—	$1/61 = 0.0164$	6th

Two half-matches beat one perfect match

Kettle articles that both lists rank fairly well collect from both. The right one, found by only one list, collects half as much and falls out of the top 5.

Fuse more, lose more

Fuse all 644 and keyword's list never ends: every chunk gets some keyword score, even pure noise. Paraphrases found: top 20, 23 of 40; top 50, 6; all, 0.

The twist: answers up, search down

Round 2: the whole system, all 520 test questions.

	SEARCH	RIGHT	NOT FOUND	LEAKS
Vector the reference	281 of 400	280 of 400	42 of 120	76 of 480
Hybrid, top 20	276 -5	299 +19	45 +3	103 +27

Gained

51 questions right that vector got wrong. Most were codes: 17.

Lost

32 the other way. Most were paraphrases and follow-ups: 9 each.

How answers rose

20 of the 51 gains had no labelled passage in the top 5. The model answered from another passage that states the same fact, which the search number does not count.

Leaks rose too: 103 against 76 of 480. Nothing stops staff-only text reaching a customer's prompt yet; the filter in E8 does.

Is +19 right answers real, or luck?

E3's quick check cannot tell

The likely ranges, 65% to 74% for vector and 70% to 79% for hybrid, overlap. When ranges overlap, look closer.

The sharper check: compare question by question

Most questions both systems get right, or both get wrong. Those tell you nothing. Only the questions where they **disagree** decide it. On all 520:

61 only hybrid right

33 only vector

If the two were equally good, the disagreements would split close to half and half, like coin flips. A split as uneven as 61 to 33 comes up by chance about **1 time in 200**.

Lesson

When ranges overlap, count the questions where the two systems disagree. Here it shows the gain is real: the hybrid gives more right answers.

A real gain, and still a no

The rule was written before the run. Checked against the results:

THE RULE	VECTOR	HYBRID	
Search clearly above vector's likely ranges must not overlap	281	276	✗
Right answers not lower	280	299	✓

Fails the first test. **The system keeps vector search.**

WHY NOT JUST CHOOSE THE BETTER-LOOKING NUMBER NOW? EACH SIDE WINS TWO:

CASE FOR HYBRID		CASE FOR VECTOR	
Right answers	299 vs 280	Search	281 vs 276
Correct "not found"	45 vs 42	Leaks	76 vs 103

Lesson

Choose the deciding number after seeing results and you can argue for either side. Fix it first, and honour it even when it says no. The gain is reported, and the season recap returns to it.

A bug found by rerunning

The symptom

Rerunning the same system changed 32 judge verdicts. The same 32 E3 saw re-judged.

The cause

With no filter, a staff member and a customer asking the same question get the same prompt. Both went to the model at once, came back worded differently, and the cache kept only one reply.

The fix

Each distinct prompt is now sent once, and its reply given to every question that made it.

The effect

The reference now reruns exactly. It scores **280** right, one fewer than E4's 281.

Why it matters

A result you cannot reproduce cannot be compared. Rerun your reference before you trust a difference of a few answers.

What this does not prove

- **One embedding model and one way of splitting words.**
Keyword search here reads "E-2194" as "e" and "2194". Kept whole, codes would behave differently again.
- **One way to merge**, with equal weight. A hybrid that trusts keyword search only when the question contains a code was not tested.
- **The dev questions have no codes or paraphrases**, so all three fuse depths tied there, and the rule's tie-break chose.
- **The search number counts only the labelled passage**, so it undercounts a search that finds the same fact elsewhere.
- **The judge** is gemini-3.6-flash, checked against E3's sample.

Run it yourself

All five searches, both rounds.

```
# rebuild every result and chart: no key, no cost
$ uv run lab rag analyse episodes/s2-e5-search/config.yaml

# round 1 on dev, then the full run (keys needed)
$ uv run lab rag run episodes/s2-e5-search/config.dev.yaml
$ uv run lab rag run episodes/s2-e5-search/config.yaml
```

Cost

\$1.48 for the hybrid's full run. The search-only rounds cost nothing new: the embeddings were cached.

Know what your users ask

Vector search is the default in most RAG tools. Here it missed 25 of the 40 questions about error codes.

Ask your team

Do our users ask with codes, part numbers or product names?
How does our search do on just those?

Ask your team

Do we score search on each kind of question, or only on the overall number?

Ask your team

Did we decide what counts as better before we ran the test?

Stage 5 tested. Not joined.

1 Load	2 Chunk	3 Embed	4 Rewrite	5 Search	6 Filter	7 Rerank	8 Prompt	9 Answer	10 Cite	11 Score
-----------	------------	------------	--------------	-------------	-------------	-------------	-------------	-------------	------------	-------------

	SEARCH	RIGHT	NOT FOUND	LEAKS
After E4 structure 1,024	281 of 400	281 of 400	42 of 120	76 of 480
Hybrid, top 20 tested, not joined	276 of 400	299 of 400	45 of 120	103 of 480
After E5 vector kept, rerun	281 of 400	280 of 400	42 of 120	76 of 480

Lime: stages that have joined. Coral: tested, not joined. The bottom row carries forward; 280 not 281 is the bug fix, not a change.

Decision rule

Test search on each kind of question your users ask. Keep keyword search for exact codes, and merge rankings only where your tests show it helps.

Next: S2 E6 Rewriting the question

"And the Tessa dishwasher?" Every search failed follow-ups like this. A model rewrites them into full questions first.

Where this fits

The series map

S1 LLMs	S2 Retrieval	S3 Strategy
S4 Adapting	S5 Agents	S6 Economics
S7 Data	S8 Evaluation	S9 AI teams
S10 Reliability	S11 Security	S12 Delivery
S13 Performance	S14 Resilience	S15 Governance

☐ Leadership season

 **Follow**

linkedin.com/in/brendan-kelly-irl

 **Episode guide**

brendankellyai.github.io

 **Code**

github.com/BrendanKellyAI/applied-ai-lab