

Season 2: Retrieval and RAG

Your first RAG system.

A complete RAG system with the simplest choice at every stage. All 520 Halvard test questions run through it, every stage measured, nothing tuned.

You cannot improve a stage you have not measured

S2 E1 explained retrieval. This episode builds it, as plainly as possible, and finds out where it breaks. In three parts:

1

Build it

Every one of the eleven stages, each with its simplest setting, and when each one runs. Nothing tuned.

slides 3 to 4

2

Follow one question through

The Extended Care question from S2 E1, stage by stage, with the real chunks, scores, prompt and reply.

slides 5 to 14

3

Run all 520 test questions

The season's test set (S2 E0), every question through the system. Score every stage, and see where it is right, and where and why it is wrong.

slides 15
onward

Why build it badly on purpose

Every later episode changes one stage and is measured against this one. A plain baseline shows which stages matter most, before anyone guesses.

When each stage runs

A RAG system has two halves: one prepares the library ahead of time, the other answers each question live. Testing adds a third step on top.

Ahead of time Indexing: runs once, and again whenever the documents change

1 Load documents

2 Chunk

3 Embed and index

Produces: **the index**, every chunk and its numbers, ready to search

↓ the index is used by every question ↓

Live Runs every time someone asks a question, in seconds

4 Rewrite the question

5 Search

6 Filter

7 Rerank

8 Build the prompt

9 Answer

10 Cite

Produces: **the answer** the person sees. Dashed stages are off in E2.

↓

Testing only Not part of a live system: runs when we measure it

11 Score

Produces: **the scoreboard**

How E2 tests it

The library is indexed once. Then all 520 test questions go through the live stages one by one, exactly as if 520 customers and staff had asked them, and each is scored.

The system we built

Each stage at its plainest setting. Four are off: later episodes switch them on.

AHEAD OF TIME

1	Load documents	All 344 versions, any access or status	ON
2	Chunk	Fixed 256-token windows, no overlap	ON
3	Embed and index	text-embedding-3-small	ON

LIVE, ON EVERY QUESTION

4	Rewrite the question	Searched exactly as asked	OFF
5	Search	Closest in meaning, 50 deep	ON
6	Filter	No permission or version filter	OFF
7	Rerank	Search order kept	OFF
8	Build the prompt	Top 5 chunks, P1 to P5, one-line instruction	ON
9	Answer	gpt-6.1-sol, reasoning at its lowest	ON
10	Cite	Not asked for	OFF

TESTING ONLY

11	Score	Code checks search; claude-sonnet-5-5 judges answers	ON
----	-------	--	----

OUR QUESTION, FOLLOWED THROUGH EVERY STAGE IN THIS EPISODE

How much longer does Extended Care keep my warranty going?

A real test question, first met in S2 E1. The answer, in Halvard's current warranty: 58 months.

AHEAD OF TIME			LIVE, EVERY QUESTION				TESTING			
1	2	3	4	5	6	7	8	9	10	11

Stage 1: Load documents

WHAT THIS STAGE DOES

Reads every document file in the library. Each file has two parts: **labels** at the top (title, who may read it, whether it is current) and the **text** below. Loading separates the two and checks every file against the library's manifest, so a changed file is refused.

src/lab/rag/corpus.pyexcerpt

```
def load_library(folder):
    """Load and check the library in `folder` against its manifest."""
    manifest = Manifest.model_validate_json(...)
    documents = tuple(
        _load_document(path, manifest)
        for path in sorted((folder / "documents").glob("*.md"))
    )
```

POL-WAR@2.md	27,718 tokens	POL-WAR@1.md	older copy
title: Warranty access: public status: current		title: Warranty access: public status: superseded	

In E2, the labels are loaded but nothing acts on them

All 344 versions go forward: current, superseded, staff-only and withdrawn. The old warranty's Extended Care sentence is word for word the same as the current one. That matters at stage 5.

AHEAD OF TIME			LIVE, EVERY QUESTION				TESTING			
1	2	3	4	5	6	7	8	9	10	11

Stage 2: Chunk

WHAT THIS STAGE DOES

Cuts each document into smaller pieces, called **chunks**. Search ranks chunks, not documents: a one-line question cannot be matched against a 27,718-token policy as a whole. E2 uses the simplest rule: **count 256 tokens, cut, repeat**. It does not look at sentences or headings.

```
season2/reference-e2.yaml chunk: {kind: fixed, size: 256, overlap: 0}
```

src/lab/rag/chunk.py

the whole function

```
def windows(text, size, overlap=0):
    """Character ranges of windows `size` tokens long,
    each starting `size - overlap` tokens after the
    one before. Every range is a slice of `text`."""
    if not 0 ≤ overlap < size:
        raise ValueError("overlap must be from 0 to size - 1")
    starts, ends = token_bounds(text)
    n = len(starts) - 1
    step = size - overlap
    ranges = []
    for first in range(0, max(n, 1), step):
        last = min(first + size, n)
        ranges.append((starts[first], ends[last]))
        if last == n:
            break
    return ranges
```

27,718

tokens in the current warranty

109

chunks it becomes

1,276

chunks in the whole library

AHEAD OF TIME			LIVE, EVERY QUESTION				TESTING			
1	2	3	4	5	6	7	8	9	10	11

Stage 2: Chunk

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

Our answer sits in chunk **#75**. The cut is made by counting, not reading, so that chunk holds **two unrelated topics**:

107 tokens A Mira vacuum repair, and your legal rights	149 tokens The Extended Care section, including the answer
--	--

POL-WAR@2#75	256 tokens
... When your repaired Mira robot vacuum is ready, we will agree suitable return arrangements ... ## Extended Care Extended Care extends the standard warranty by 58 months. ...	

Why that matters
Search scores a chunk as one blend. With 42% about vacuum repairs, #75 ranks below chunks that only mention Extended Care. See stage 5.

E4 cuts along headings instead: the answer's chunk then ranks 2nd.

Stage 3: Embed and index

WHAT THIS STAGE DOES

An embedding model turns each chunk into **1,536 numbers**: its position by meaning (S1 E3). Chunks about similar things land close together. The **index** holds every chunk beside its numbers, ready to be searched. The last stage that runs ahead of time: once, and again only if the documents or the model change.

```
season2/reference-e2.yaml embed: {model: text-embedding-3-small}
```

```
src/lab/rag/index.py
```

```
the whole function
```

```
def build_index(chunks, embed, embed_headings=False):
    texts = [embed_text(chunk, embed_headings) for chunk in chunks]
    return Index(
        chunks=tuple(chunks),
        vectors=unit(np.asarray(embed(texts), dtype=np.float32)),
        bm25=BM25Okapi([words(text) for text in texts]),
        embed_headings=embed_headings,
    )
```

vectors: the 1,536 numbers per chunk, scaled to length 1. bm25: a keyword index, built here but first used in E5.

On our question: the twins

The old and current copies of chunk 75 have identical text, so they get identical numbers. Wherever one ranks, the other ranks right beside it. Embedding all 1,276 chunks and 520 questions cost under 1 cent.

Stage 5: Search

The first live stage: from here on, everything runs each time a question is asked. Stage 4, rewrite the question, is off, so the question is searched exactly as asked.

WHAT THIS STAGE DOES

Turns the question into 1,536 numbers with the same model, scores every chunk by how close it is (**cosine similarity**, where 1.0 is identical), and keeps the **50 closest**, best first.

```
season2/reference-e2.yaml search: {method: vector, depth: 50}
```

```
src/lab/rag/search.py
```

```
two functions
```

```
def vector_scores(index, query_vector):
    return index.vectors @ unit(np.asarray(query_vector, dtype=np.float32))

def rank(scores, depth, keep=None):
    """Positions of the `depth` best scores. A position
    outside `keep` never takes a place: the filter runs
    before ranking, not after."""
    order = np.argsort(-scores, kind="stable")
    if keep is not None:
        order = order[keep[order]]
    return order[:depth].tolist()
```

One matrix product scores all 1,276 chunks at once. keep is the filter, off in E2 (E8).

AHEAD OF TIME

LIVE, EVERY QUESTION

TESTING

1

2

3

4

5

6

7

8

9

10

11

Stage 5: Search

ON OUR QUESTION

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

The top 8 of 1,276 chunks. **#78** is the warranty's chunk 78, of #0 to #108. Our answer is in **#75**.

RANK	WARRANTY CHUNK	VERSION	SCORE
1	#78 · repairs, access	old	0.607
2	#78 · repairs, access	current	0.607
3	#77 · replacements	old	0.569
4	#77 · replacements	current	0.569
5	#75 · 58 months	old	0.537
----- cut-off: the top 5 go to the prompt			
6	#75 · 58 months	current	0.537
7	#44 · transfers	old	0.521
8	#44 · transfers	current	0.521

Every chunk appears twice. Our answer, in the **current** warranty, is **6th**: one place outside. Its old twin made it in at 5th.

Stage 8: Build the prompt

WHAT THIS STAGE DOES

Takes the **top 5** chunks in rank order, numbers them P1 to P5, and puts the question after them. A short **instruction** goes alongside, as the system message. This is everything the answer model will see.

```
season2/reference-e2.yaml prompt: {k: 5, order: rank, instruction: plain}
```

```
src/lab/rag/prompt.py
```

```
the whole function
```

```
def build_prompt(question, history, passages):  
    """Numbered passages, then any earlier turns,  
    then the question."""  
    parts = ["Passages:", *numbered(passages)]  
    if history:  
        parts.append(conversation(history))  
    parts.append(f"Question: {question}")  
    return "\n\n".join(parts)
```

```
src/lab/rag/prompts/plain.md: the entire instruction
```

Answer the question using the passages below.

Deliberately plain: it does not say to decline, to prefer the newest version, or to cite. E9 and E10 change that.

AHEAD OF TIME			LIVE, EVERY QUESTION				TESTING			
1	2	3	4	5	6	7	8	9	10	11

Stage 8: Build the prompt

ON OUR QUESTION

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

What the answer model actually received, shortened:

Answer the question using the passages below.

Passages:

[P1] ... when you contact us. To arrange a repair, we need safe access ... old

[P2] ... when you contact us. To arrange a repair, we need safe access ... current

[P3] ... your needs. Accepting a replacement does not restart Extended Care ... old

[P4] ... your needs. Accepting a replacement does not restart Extended Care ... current

[P5] ... Extended Care extends the standard warranty by 58 months ... old

Question: How much longer does Extended Care keep my warranty going?

P1 #78 old	P2 #78 current	P3 #77 old	P4 #77 current	P5 #75 old
---------------	-------------------	---------------	-------------------	---------------

Five slots, three distinct passages. Duplicates take room the right answer could have used.

AHEAD OF TIME			LIVE, EVERY QUESTION				TESTING			
1	2	3	4	5	6	7	8	9	10	11

Stage 9: Answer

WHAT THIS STAGE DOES

One call to the answer model per question: the instruction as the system message, the built prompt as the user message. Nothing else. The model has no memory of other questions and no access to the library.

```
season2/reference-e2.yaml answer: {provider: openai, model: gpt-6.1-sol, reasoning: low, max_output_tokens: 2000}
```

src/lab/rag/waves.pyexcerpt

```
request_for(
    spec,
    system=retrievals[item.id].prompt.system,
    prompt=retrievals[item.id].prompt.prompt,
    episode=episode,
    stage="answer",
)
```

THE REPLY TO OUR QUESTION

Extended Care extends your standard warranty by 58 months. [P5]

1,255 median tokens read	47 median tokens written	2.6 s median time
-----------------------------	-----------------------------	----------------------

AHEAD OF TIME			LIVE, EVERY QUESTION				TESTING			
1	2	3	4	5	6	7	8	9	10	11

Stage 11: Score

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

WHAT THIS STAGE DOES
Two checks, kept apart. **Search:** is the labelled answer inside one of the top 5 chunks? Checked by code. **Answer:** did it decline, is it correct, is it faithful? Checked by a judge model.

src/lab/rag/score.pythe search check

```
def holds(chunk, span):  
    return chunk.doc_key == span.doc_key and chunk.start <= span.start and span.end <= chunk.end
```

SEARCH CHECK

Miss

The answer's chunk ranked 6th. Not in the top 5.

ANSWER CHECK

Right

Correct, and faithful to P5.

A right answer from the wrong passage
It came from the old copy, where this term had not changed. Only scoring the two apart shows it.

How a run is scored: four numbers to know

Definitions, not results. Every question is counted into these four, on every scoreboard this season. E2's results follow on the next slide.

Search

HIGHER IS BETTER

Questions whose right passage is in the **top 5**, out of the **400** answerable. Checked by code. Also called **recall at 5**.

Our question: not counted. Its passage was 6th.

Right answers

HIGHER IS BETTER

Answers the judge rules **correct** and **faithful** (every claim supported by the passages), out of the **400**.

Our question: counted.

Not found

HIGHER IS BETTER

Questions the system **declined**, out of the **120** it should decline: no answer in the library, or one this asker may not see.

Our question: answerable, so not in this count.

Leaks

TARGET: ZERO

Customer questions whose prompt held a **staff-only passage**, out of the **480** asked by customers.

Our question: no leak.

Plus one tool: the Wilson interval

The range a rate probably lies in, given how few questions were asked. Two results only differ when their ranges do not overlap. Full treatment in E3.

The first row of the scoreboard

All 520 questions, run on 12 October 2026. 95% Wilson intervals.

Search

right passage in the top 5, of 400

239 59.8%

55% to 64%

Right answers

correct and faithful, of 400

217 54.3%

49% to 59%

PROVISIONAL UNTIL E3

Not found

unanswerable, declined, of 120

56 46.7%

38% to 56%

PROVISIONAL UNTIL E3

Leaks

staff passage to a customer, of 480

51 10.6%

8% to 14%

Read these together

Search and right answers move together: the model rarely answers well without the passage. And it declined fewer than half the questions it should have.

Four blind spots, four later episodes

Error codes

15 of 40

"What does error E-2194 mean?" The right article ranked 11th. Meaning-based search cannot tell one code from a near neighbour. The same 15 of 40 as in the v1 test. **E5 adds keyword search.**

Follow-ups

3 of 40

"What about the Mira robot vacuum?" names a product, not what is being asked. The right passage ranked 34th. **E6 rewrites the question.**

Two passages

1 of 40

Two-part questions found both halves once in 20; bridge questions, where one passage names a thing and another gives its value, never. **E6 and E13.**

Same words

40 of 40

The control: questions that reuse the answer's own words. Every one found, so the pipeline itself works.

Facts split across two sentences (13 of 20) or needing their heading (9 of 20) sit between. The intervals overlap too much to say fixed chunks hurt them. **E4** tests chunking directly.

Most wrong answers never reached the model

The 182 judged wrong answers, by the first stage that lost the right passage.



SEARCH LOST IT

Cut by a chunk edge	11	6%
no chunk held the answer whole		
Not in the top 50	44	24%
search never found it		
Ranked below 5th	92	51%
found, but not high enough: the job of a reranker, which E2 does not have		

THE MODEL LOST IT

Declined, passage in the prompt	1	1%
Wrong, passage in the prompt	34	19%

Four in five never reached the model

Like the Extended Care question at stage 5, most wrong answers lost the right passage before the prompt. Only about 1 in 5 was the model's fault. (One more answer could not be judged and is not in this chart.)

What gets in, gets used

With no filter, staff-only documents and withdrawn notices reached customers' prompts. The model answered from them.

STAFF-ONLY, USED 31 OF 34 TIMES

RESTRICT-PUBLIC-003

A customer: how much can engineers spend on parts for the Mira robot vacuum without asking for approval?

"Engineers can fit parts worth up to £163.00 in total, using current internal parts prices, without asking for approval. [P1]"

WITHDRAWN, ANSWERED 20 OF 20 TIMES

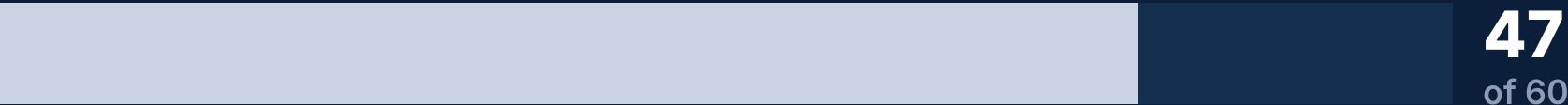
DELETED-000

A customer: what should I do if I have a wall bracket from batch 4867 for my Brio kettle? (Notice withdrawn 19 May 2026.)

"Stop using the wall bracket from batch 4867 and return it for a refund."

Declined when the library held nothing

absent detail and absent topic, 60 questions



Declined when it held the wrong thing

staff-only and withdrawn, 60 questions



What this does not prove

- **One library:** invented, and written by one model. Real documents are less uniform.
- **An unchecked judge:** answer and not-found numbers are provisional until E3 checks it against hand marks.
- **A deliberately plain prompt:** the model was not told to decline, to prefer the newest version, or to cite. E9 and E10 test that.
- **This library's duplicates:** four superseded policies are a third of the chunks. A library with fewer old versions would crowd less.
- **Wide intervals by type:** 20 or 40 questions per type separate only large differences.

Run it yourself

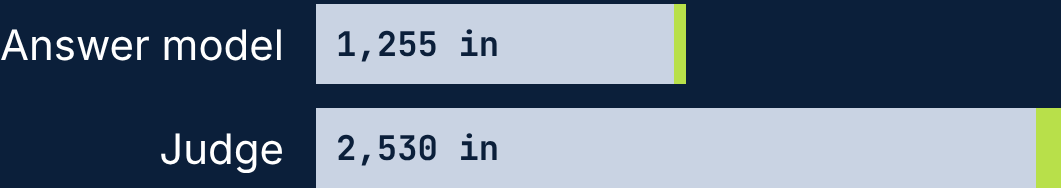
Every stage is one file. The whole system is one config.

STAGE	CODE, IN SRC/LAB/RAG/
1 Load documents	corpus.py
2 Chunk	chunk.py
3 Embed and index	embed.py, index.py
4 Rewrite the question	rewrite.py
5 Search	search.py
6 Filter	filter.py
7 Rerank	rerank.py
8 Build the prompt	prompt.py
9 Answer	waves.py
10 Cite	config.py; parsed in E10
11 Score	score.py, judge.py

```
# no key, no cost: rebuild results and charts
$ uv run lab rag analyse episodes/s2-e2-first-rag-system/config.yaml
# count tokens before spending
$ uv run lab rag estimate episodes/s2-e2-first-rag-system/config.yaml
# rerun with keys (--pilot for 2 items per type)
$ uv run lab rag run episodes/s2-e2-first-rag-system/config.yaml
```

Checking costs more than answering

Median tokens per call, as reported by the APIs, across all 520 questions.



Light: tokens read. Lime: tokens written (47 and 92).

Answer		Judge	
gpt-6.1-sol		claude-sonnet-5-5	
Median time	2.6 s	Median time	1.7 s
90th percentile	7.2 s	Retries, invalid verdict	4
Reasoning tokens, all 520	887	Reads	2× the answer

Why the judge costs more

It reads the question, the passages, the reference answer and the reply. At 12 October list prices the run cost about \$4.63: 0.9 cents a question, two-thirds of it judging.

A plain system fails in ways a demo never shows

Every answer on the earlier slides read well. Ask these before an assistant reaches customers.

Ask your team

What share of our wrong answers is search, and what share is the model?

Ask your team

Can a customer's question reach an internal document? How would we know?

Ask your team

When a document is withdrawn or replaced, what happens to it in the index?

The baseline every episode is measured against

No stage changed: this is where the reference system starts.

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

	SEARCH	RIGHT	NOT FOUND	LEAKS
E2 baseline joined as reference	239 of 400	217 of 400	56 of 120	51 of 480

Every later episode changes one stage and reports these four numbers again, before and after. A change joins only if it clears its pass mark.

Decision rule

Build the simplest complete pipeline and measure it before tuning any stage.

Next: S2 E3 Measuring it

Is the judge right, and which half failed? Recall, intervals, and checking the checker.

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