

Season 2: Retrieval and RAG

Chunking.

How you cut a document decides what search can find. Fifteen ways to cut Harvard's library, tested against the 520 test questions.

One stage changes: the cut

The first episode to change the system. Everything else stays exactly as E2 built it.

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

E2 cut every document into fixed 256-token pieces, wherever the count ran out. Three things it showed:

Mixed topics	Our Extended Care answer shared a chunk with a vacuum repair. It ranked 6th, one place outside the prompt.
Cut facts	On 18 questions, a chunk edge cut the answer in two, so no chunk held it whole.
Stitched sentences	In E3, a model had to rejoin a sentence split across two chunks.

The question

Is there a better way to cut each document into chunks, so search finds the passage holding the answer more often? And if it does, do we get more right answers?

Two words to keep apart: token and chunk

TOKEN: THE UNIT OF MEASURE (S1 E2)

A model reads text as tokens, pieces of about three quarters of a word. Our answer sentence is **11 tokens**:

Extended	Care	extends	the	standard	warranty	by		58
months	.							

Tokenising is fixed by the model. We do not change it.

CHUNK: THE PASSAGE SEARCH RETURNS (S2 E2)

A chunk is a stretch of a document, **hundreds of tokens long**. Search ranks chunks, and the top 5 go into the prompt. Its size is counted in tokens: E2's chunks were 256 each.

chunk #74 · ... 256 tokens ...

chunk #75 · ... Mira vacuum repair ... ## Extended Care ... extends the standard warranty by 58 months ...

chunk #76 · ... 256 tokens ...

What stage 2 decides

Where each chunk starts and ends. The words stay the same and the tokens stay the same; only the cuts move.

Three ways to cut, tested here

A policy has sections under headings. Shown here: three sections, and the chunks each method makes. There are more ways: next slide.

Fixed window Count N tokens, cut, repeat. Ignores the text.	Section ASection BSection C
Fixed, with overlap Each window starts before the last one ends.	Section ASection BSection C
Structure One chunk per section, split at paragraphs only if too long.	Section ASection BSection C
Coral	Fixed windows straddle sections: one chunk, two topics.
White	Overlap means a fact cut by one window is whole in the next.
Lime	Structure chunks follow the document's own sections: one chunk, one topic.

Other ways to cut, not tested here

Chunking is an open field. These are the common alternatives.

Sentence or recursive

e.g. LangChain's default splitter

Cut at the largest natural break that fits: paragraph, then sentence, then word. Close to our structure cut, without reading headings.

Semantic

Embed every sentence and cut where the meaning shifts between neighbours. Costs an embedding per sentence before search starts.

Small to big

parent and child

Search small chunks for precision, then send the larger section around the match to the model.

Model-written

contextual, propositions

A language model rewrites or labels each chunk before it is embedded. The most expensive: a model call per chunk.

Is semantic a big miss?

Not for this library. Semantic chunking looks for where the topic changes; Halvard's headings already mark exactly that, and the structure cut split no answer at all. Published tests found semantic chunking's gains inconsistent for its cost (Qu et al., NAACL 2025). It matters most for text **without** headings: transcripts, emails, scanned pages.

Stage 2: Chunk

WHAT CHANGES IN THIS STAGE

The structure chunker finds every heading, makes **one chunk per lowest-level section**, and only splits a section if it is longer than the limit, at paragraph breaks. Halvard's policy sections are about 840 tokens, so at a limit of 1,024 every section stays whole.

```
E2 chunk: {kind: fixed, size: 256, overlap: 0}
```

```
E4 chunk: {kind: structure, max_tokens: 1024}
```

src/lab/rag/chunk.py

trimmed: full code in the repo

```
def _fit(body, start, end, max_tokens):
    """The section as one range if it fits,
    else split at paragraph breaks, else by tokens."""
    # 1. The whole section fits: one chunk
    if count_tokens(body[start:end]) ≤ max_tokens:
        return [(start, end)]

    # 2. Too long: gather paragraphs until the next would not fit
    for a, b in paragraphs:
        ...

    # 3. One paragraph alone too long: fixed windows inside it
    ranges.extend(windows(body[a:b], max_tokens))
```

Labels travel

Every chunk copies its document's labels: who may read it (**access**) and whether it is current (**status**). Cutting never checks them; the filter in stage 6 does, in E8.

Our question, cut two ways

OUR QUESTION

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

E2: FIXED 256

Chunk #75, two topics

... When your repaired Mira robot vacuum is ready ...
Extended Care
Extended Care extends the standard warranty by 58 months. ...

Size	256 tokens
Score	0.537
Rank	6th

Outside the top 5

E4: STRUCTURE 1,024

One section, one topic

Warranty > Extended Care > What the plan adds
Extended Care extends the standard warranty by 58 months. It adds protection against covered faults ... (the whole section)

Size	858 tokens
Score	0.586
Rank	2nd

In the prompt

Longer, but on one subject

The new chunk is three times longer, yet it scores higher: every word in it is about Extended Care. Search now finds it, and the answer is right from the current warranty.

The whole library: 1,276 fixed chunks become 644 sections. The current warranty: 109 become 34.

Fifteen cuts, tested in two rounds

15 New ways to cut, each a new stage 2

Fixed windows at five sizes, each with and without overlap (one of those ten is E2's own cut, the reference). Structure at three limits, each with and without the heading path added.

64	128	256	512	1,024	+25% overlap	structure 256	512	1,024
+ headings								

1 Round 1: search only, all fifteen

Search is checked by code, so it needs no answer model or judge. Cheap enough to try every cut on all 400 answerable questions.

2 Round 2: the full system, the best two

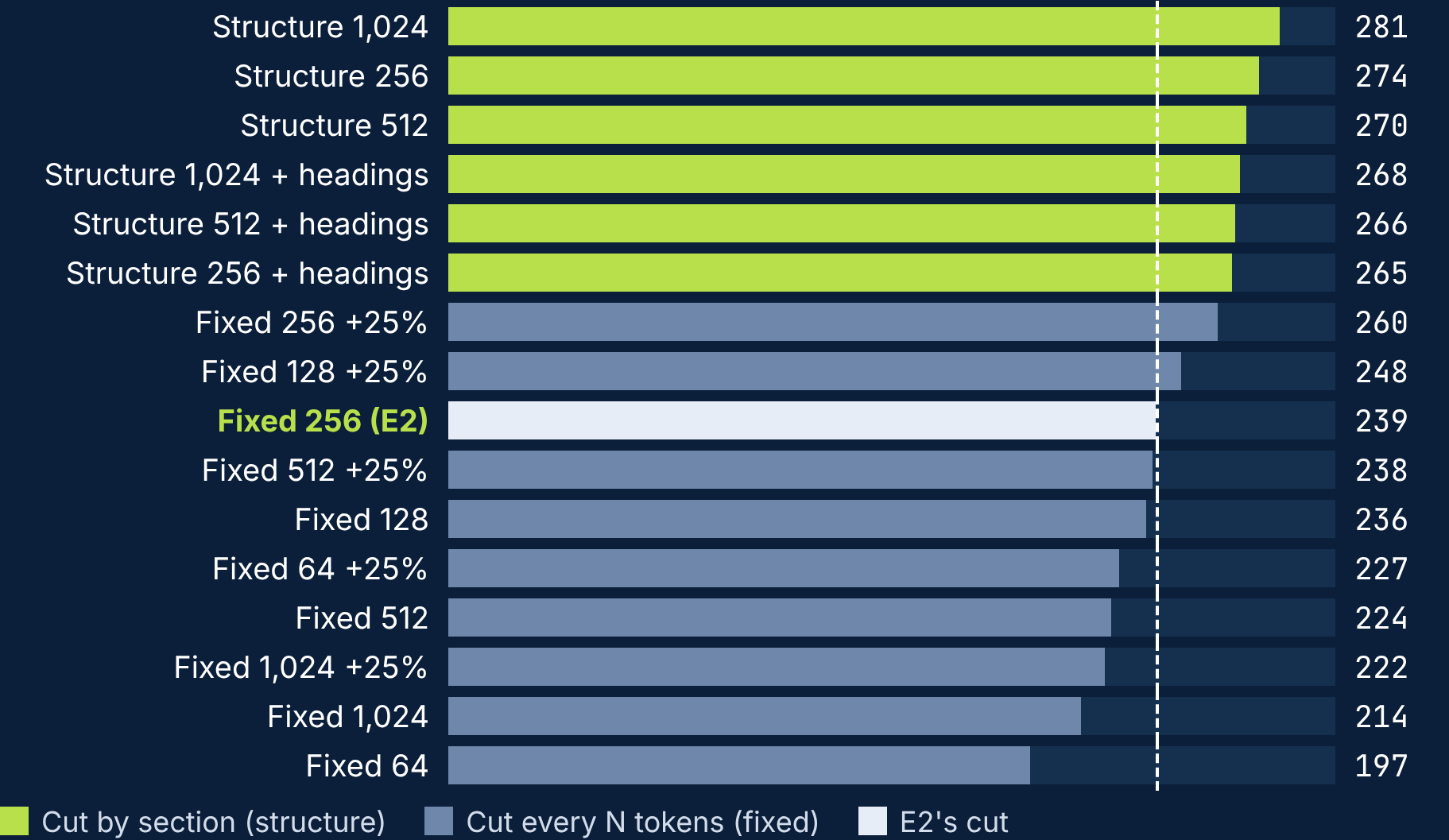
The best fixed and the best structure cut, picked on a separate set of 100 dev questions, then run end to end on the 520 test questions.

Why a separate dev set

Choosing on the same questions you report on flatters the winner. Halvard keeps 100 dev questions for choosing, and the 520 test questions for the result.

Cutting by section beat every fixed size

Each bar is one way of cutting the same library, 16 in all. **Its length:** of the 400 questions that have an answer, how often search put the passage holding it in the top 5, the ones the model sees.



How to read it

All six lime bars are longer than all ten grey ones. The best, structure 1,024, found the answer for **281** questions; E2's cut found **239**. That is 42 more.

How a fixed window cuts an answer in two

A REAL TEST QUESTION

How much does it cost to get the Corra induction hob serviced? Answer: £210.00

E2 counts 256 tokens and cuts, wherever it lands. Here it landed in the middle of the price:

CHUNK #56 ENDS

... Please tell us about access restrictions when arranging the appointment. A routine service of the Corra induction hob costs £

Has the question's words. **No price.**

CHUNK #57 STARTS

210.00. We will explain the payment arrangements when you book. ...

Has the price. **No idea what it is for.**

Search can return either chunk, but **neither one answers the question.**

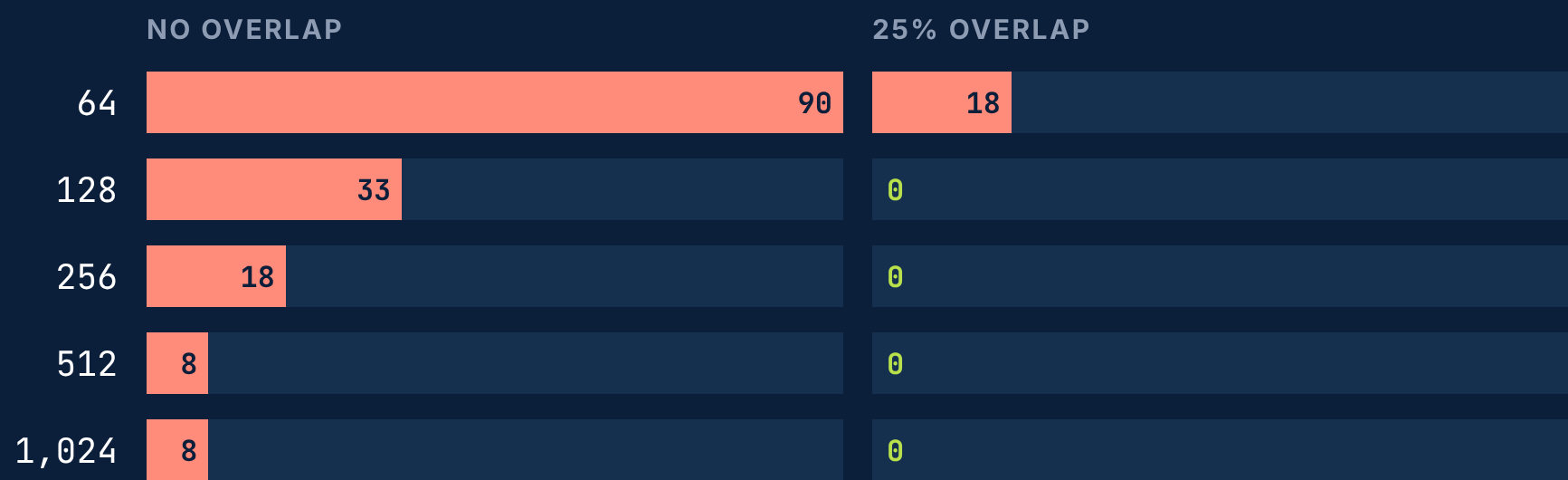
THE FIX: OVERLAP. EACH WINDOW STARTS 64 TOKENS BEFORE THE LAST ONE ENDED

... access restrictions when arranging the appointment. A routine service of the Corra induction hob costs £210.00. We will explain the payment ...

The repeated 64 tokens carry the start of the sentence into the next window. **One chunk now holds the whole answer.**

Overlap stops answers being cut in two

Of the 400 questions, how many had their answer cut by a window edge, like the Corra hob price. Rows are window sizes, in tokens.



Why 64 still cuts

A 64-token window repeats only 16 tokens, shorter than many answer sentences.

Cut by section

Cut no answers at all. Sections end where a topic ends, never mid-sentence.

So why not just add overlap?

It helps: fixed 256 with overlap was the best fixed cut (260). But it fixes cut sentences only. Chunks still mix topics, like our chunk #75, so cutting by section still won.

Two common ideas that did not hold

COMMON ADVICE

"Long chunks dilute a fact, so keep them small."

```
Policy facts found: fixed 1,024: 48 of 80 · fixed 256: 46 of 80
```

No dilution here. Earlier tests hid one invented fact in unrelated novel text, where extra words are noise. In a policy, the text around a fact is **about the same subject**, so a longer chunk adds related words.

COMMON ADVICE

"Add the heading path to each chunk before embedding it."

```
Facts that need their heading, of 20: 14 and 14 · 12 and 11 · 15 and 15
```

Without and with headings, at each size: no difference. A structure chunk already **starts with its heading**, so the product name is in the text anyway.

Test advice on your own documents. Both ideas are true somewhere; neither was true here.

Better search, better answers

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

	SEARCH: ANSWER IN THE TOP 5	RIGHT ANSWERS
E2's cut fixed 256	239 of 400 · 60%	243 of 400 · 61%
New cut structure 1,024	281 of 400 · 70%	281 of 400 · 70%

Search found the answer for **42 more questions**, and the system gave **38 more right answers**.

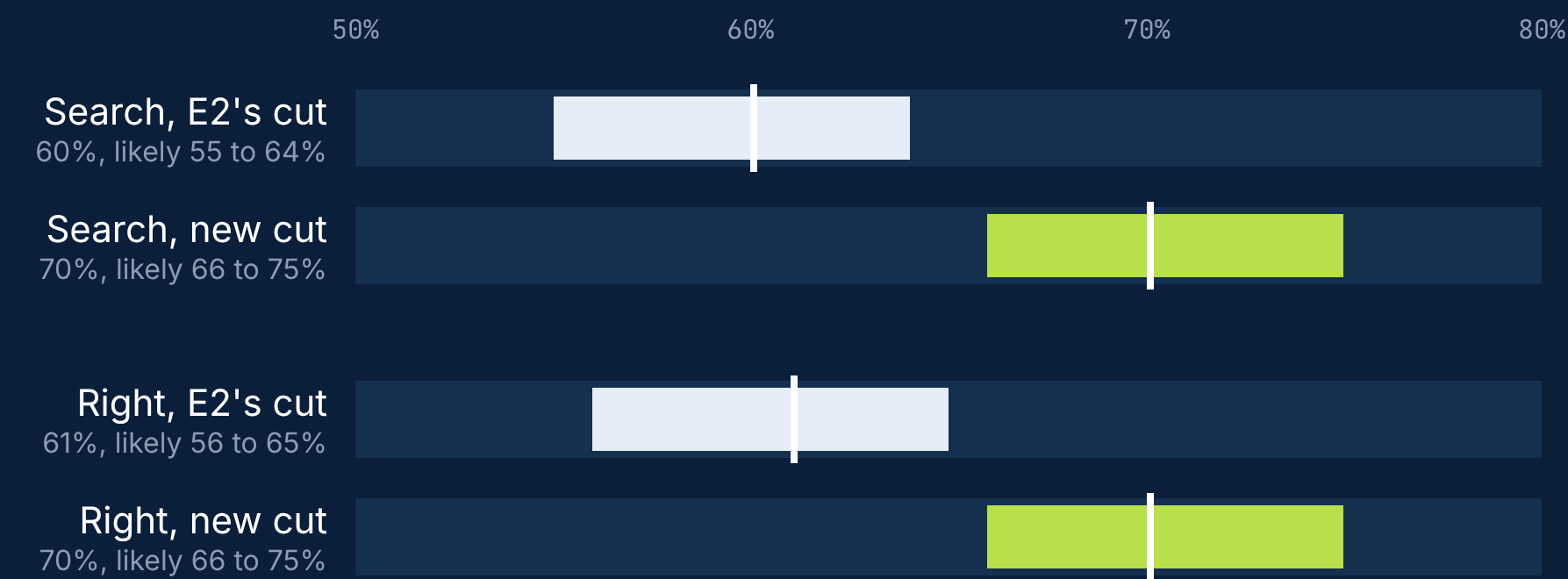
Why search improved	Answers are no longer cut in two, and each chunk is about one subject, so it matches the question more closely.
Why answers improved	The model can only answer from what it is given. More right passages in the prompt, more right answers.
But is it real?	Next slide: E3's check.

Real, or luck?

Could a different 400 questions have shown no gain? E3's check answers that.

The likely range

A score from 400 questions is an estimate. The Wilson interval is the range the true rate probably sits in. Below, each bar is that range and the white mark is the score. **If two bars do not overlap, the difference is real.**



No overlap on either

The gain is real on search and on right answers. The new cut joins the system.

All four numbers, three systems

Round 2 ran three versions: E2's cut, and the best fixed and best structure cut picked on the dev questions.

	SEARCH	RIGHT	NOT FOUND	LEAKS
Fixed 256 E2's cut, the reference	239 of 400	243 of 400	53 of 120	51 of 480
Fixed 512 best fixed on dev	224 of 400	237 of 400	57 of 120	30 of 480
Structure 1,024 best structure on dev	281 +42	281 +38	42 -11	76 +25

Search, Right	Higher is better. Of the 400 questions that have an answer.
Not found	Higher is better. Of the 120 with no answer, how many the system correctly declined.
Leaks	Lower is better. Staff-only text in a public asker's prompt, of 480.

The reference scores 243 right, not E2's 217, because the answer model changed this episode (two slides on). Search does not use that model, so it stays at 239.

What the better cut made worse

A change can help one number and hurt another. Every episode reports all four.

Leaks 51 → 76 of 480	A whole staff-only section now arrives as one chunk that matches its topic well, so it reaches customers' prompts more often.	E8
Not found 53 → 42 of 120	Longer chunks give the model more nearby text to answer from when it should decline.	E9
Prompt size 1,173 → 2,914	Median tokens per prompt, about 2.5 times longer : every answer and every judgement costs more.	E9

It still joins: it clears its bar on search and right answers. The episode named on the right is where each cost is taken on.

Cheaper models, checked first

From E4 the system uses cheaper models: answers by **gpt-6-luna**, judged by **gemini-3.6-flash**. A new judge must pass E3's check before it is trusted. Four were tried, cheapest first:

CANDIDATE JUDGE	DECLINED, OF 50	CORRECT, OF 50	WITH PERSON, OF 10	
gemini-3.1-flash-lite \$0.25 / \$1.50	46	30	6	FAILED
gpt-5-mini \$0.25 / \$2.00	40	46	7	FAILED
gemini-3.5-flash-lite \$0.30 / \$2.50	46	30	6	FAILED
gemini-3.6-flash \$0.75 / \$3.75	47	49	9	PASSED

Agreement with E3's judge, and with E3's person. Pass marks: 45, 45 and 8. Prices per million tokens in and out.

Why it matters

Both Flash-Lite judges called an answer correct whenever it matched the passages, ignoring the rules for declines. The cheapest judge would have scored the season wrong. And because the answer model changed, E2's cut was rerun on the new models, so every comparison here is like for like.

What this does not prove

- **Tidy headings:** one library, written by one model. Real documents have messier structure, and structure chunking depends on it.
- **Five passages, whatever their size:** the longer cuts also sent longer prompts. E9 varies how many passages go in.
- **No semantic or sentence-level chunking** was tested.
- **Choosing on 20 dev policy questions** is noisy, and two of the choices were ties.
- **The new judge** was checked on E3's sample, not against new hand marks.

Run it yourself

All 15 chunkers, both rounds, and the judge check.

```
# rebuild every result and chart: no key, no cost
$ uv run lab rag analyse episodes/s2-e4-chunking/config.yaml

# count tokens before spending
$ uv run lab rag estimate episodes/s2-e4-chunking/config.yaml

# round 1 on dev, then round 2 on test (keys needed)
$ uv run lab rag run episodes/s2-e4-chunking/config.dev.yaml
$ uv run lab rag run episodes/s2-e4-chunking/config.yaml

# the judge check
$ uv run python episodes/s2-e4-chunking/judge_switch.py --report
```

Cost

\$3.04 for the three full runs, \$0.15 for the judge checks, about \$0.06 to embed all 15 cuts.

A chunk size is a decision, not a default

Most RAG systems ship with whatever chunk setting the library came with. Here, choosing well added 42 found passages out of 400.

Ask your team

How are our documents cut, and was that choice tested on our own questions?

Ask your team

Do our documents have reliable headings and sections we could cut along?

Ask your team

When we changed one setting, which of our other numbers got worse?

Stage 2 changed. It 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
E2 baseline original models	239 of 400	217 of 400	56 of 120	51 of 480
E2 cut, new models E4's reference	239 of 400	243 of 400	53 of 120	51 of 480
After E4 structure 1,024	281 of 400	281 of 400	42 of 120	76 of 480

Every later episode starts from the bottom row.

Decision rule

Cut documents along their own structure, then test it on your own questions.

Next: S2 E5 Search

Keyword, vector, or both. Why E-2194 is hard to find by meaning, and easy to find by spelling.

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