

Season 1: How LLMs work Field note

Lost in the middle.

Does where a fact sits in a long document decide whether a model finds it? Tested across three providers, 270 calls.

This episode is a field note

E1 to E6 explained how LLMs work. A field note takes one claim and tests it, with code you can rerun.

	Explainer episodes	Field note
Starts with	A concept to understand	A claim to test
Evidence	Diagrams and small demonstrations	A controlled experiment on models from three providers
You get	A mental model, and where it breaks	A measured result, and its limits
Code	Snippets to try	The full harness and data, rerunnable

How to read it



Season 1 has two field notes: this one, and E10 on reasoning models.

A 2023 finding still shapes how prompts are written

- **The finding:** models used information at the start and end of a long context more reliably than information in the middle (Liu et al., "Lost in the Middle")
- **How bad it got:** GPT-3.5 Turbo lost more than 20 points with the answer in the middle. At worst, it scored below its own result with no documents at all
- **The advice that followed:** put important material at the start or the end of a prompt

The shape they reported: a U

Found more often

Missed more often

Start

Middle

End

Accuracy by where the answer sat. Shape only, redrawn for illustration.

Since then, context windows have grown by orders of magnitude. Does the U still hold?

Predict before you swipe

Three current models. One invented fact, moved from the start to the end of documents up to 64,000 tokens long. What happens in the middle?

A A deep U, like 2023

B A small dip in the middle

C Fine at 4,000 tokens, a dip only at 64,000

D No dip at all

Hold your answer

Pick one, then check it on the next slide. Drop your letter in the comments.

Answer D: position did not matter

270 of 270

Three current models found the invented fact in every call: at the start, middle, and end of the document, at 4,000, 16,000, and 64,000 tokens.

95% interval: 98.6% to 100%. Slide 11 explains what that range means.

Read this with the result

This was the easiest kind of long-context task: one distinctive fact to look up. For simple lookups, position no longer matters. That is not the same as solved.

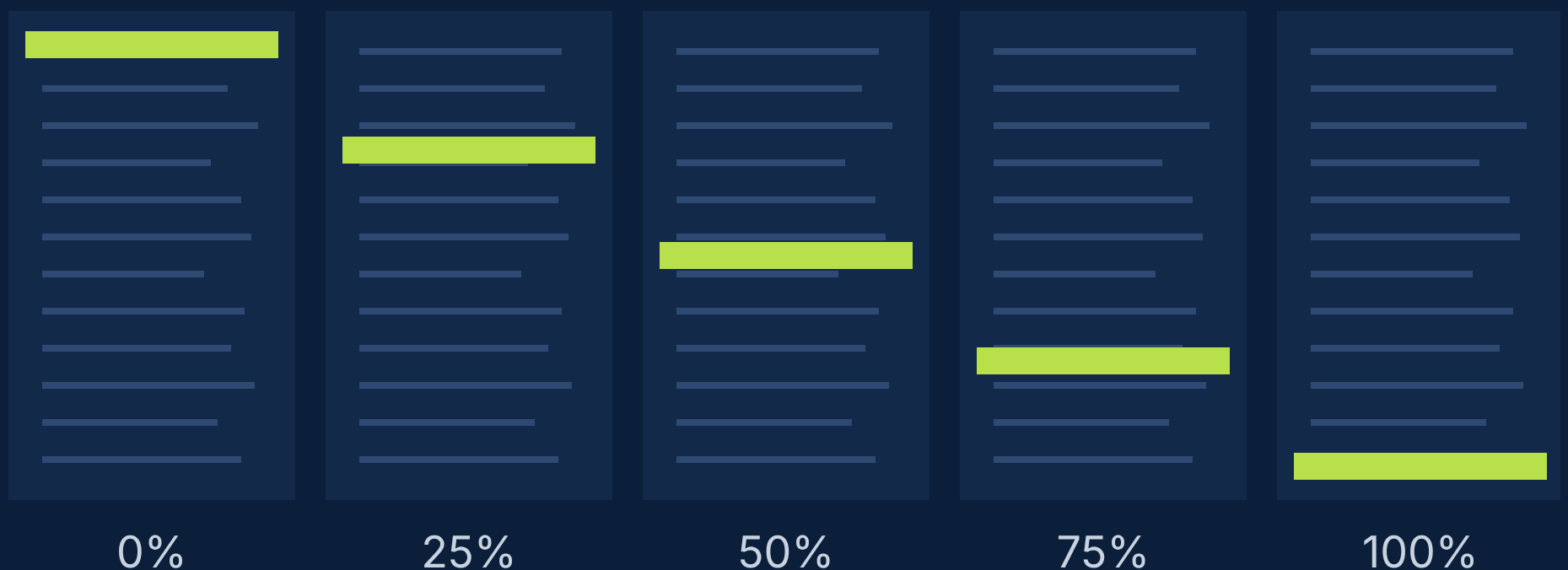
The setup

Models	GPT-5.6 Terra, Claude Sonnet 5, Gemini 3.6 Flash: each maker's balanced mid-tier model
Reasoning	Off for Terra and Sonnet 5. Gemini cannot turn thinking off, so it ran at minimal , its lowest level
Lengths	4,000, 16,000, and 64,000 tokens
Positions	0%, 25%, 50%, 75%, and 100% through the document
Filler	Six public-domain novels, one per fact, from Pride and Prejudice to Frankenstein
Calls	3 models x 3 lengths x 5 positions x 6 facts = 270

Run on 18 September 2026, 19:13 to 19:40 UTC. Every model returned the exact version requested. Provider default temperature for all three.

One invented fact, moved through the same text

The same passages carry every position and length. Only the position of the fact changes.



■ The inserted fact: the needle

Example

Inserted: "The maintenance code for turbine K-417 is 8352."

Asked: "What is the maintenance code for turbine K-417?"

This design is known as a needle-in-a-haystack test: one fact hidden in a long stretch of unrelated text.

Built to be checked

Each check removes a boring explanation for the result.

Synthetic facts	Invented turbine codes, generated from a seed. No model can know them from training, so it has to read the document
Paired design	Each fact has its own book, and that book carries it at every position and length. A dip could not come from an easier passage
Measured, not assumed	The harness checked where each fact really landed: within 0.59 points of the target position, and within 0.55% of the target length. Tolerances were 2 points and 5%
Clean insertion	Facts go in at real sentence boundaries, never inside a sentence or after "Mrs."
Pilot first	21 calls covering every model, length, and position. It caught a crash in the analysis before the full run

No dip anywhere

45 cells across three models. Each cell is six calls, one per fact. A miss would show as a darker cell. There were none.

GPT-5.6 Terra

4,000	6 of 6	6 of 6	6 of 6	6 of 6	6 of 6
16,000	6 of 6	6 of 6	6 of 6	6 of 6	6 of 6
64,000	6 of 6	6 of 6	6 of 6	6 of 6	6 of 6

Claude Sonnet 5

4,000	6 of 6	6 of 6	6 of 6	6 of 6	6 of 6
16,000	6 of 6	6 of 6	6 of 6	6 of 6	6 of 6
64,000	6 of 6	6 of 6	6 of 6	6 of 6	6 of 6

Gemini 3.6 Flash

4,000	6 of 6	6 of 6	6 of 6	6 of 6	6 of 6
16,000	6 of 6	6 of 6	6 of 6	6 of 6	6 of 6
64,000	6 of 6	6 of 6	6 of 6	6 of 6	6 of 6

tokens

0%

25%

50%

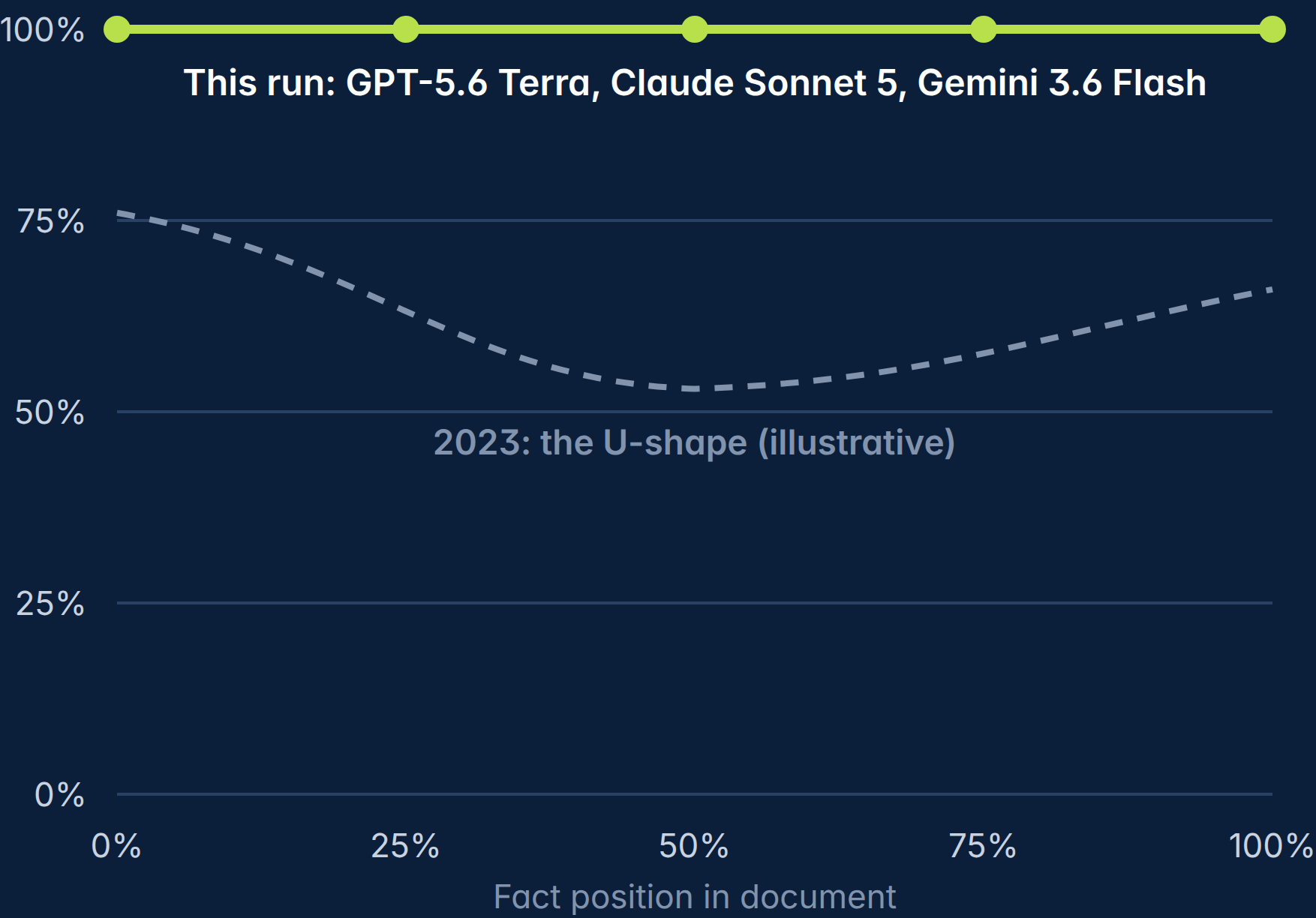
75%

100%

Fact position in document

Flat where 2023 dipped

Accuracy by position at 64,000 tokens. All three models sit on the same line at 100%, so they overlap.



Lines: n = 6 per model per point. The dashed curve is the 2023 shape, redrawn for comparison, not data from this run.

How sure is a perfect score?

A perfect score cannot show how good a model is, only a floor. The floor is the lowest true accuracy that could still plausibly produce it, and it rises as calls add up.

Calls	All correct, across	Floor
6	One heatmap cell	54.1%
54	Middle positions at 64,000 tokens	93.4%
90	One model, everything	96.0%
270	The whole run	98.6%

Floor: lower end of the 95% interval (Clopper-Pearson). If a model were truly 98.6% accurate, 270 correct in a row would happen only 2.5% of the time.

The lesson

Six out of six proves little. Judge long-context reliability on hundreds of calls, not a handful.

What this does not prove

The effect did not appear in this test. That is not evidence it has gone.

- **A perfect score has no headroom:** it cannot show whether any model was close to failing
- **The statistics rule out a large dip, not a small one:** 54 of 54 in the middle at 64,000 tokens (95% interval: 93.4% to 100%). Any middle dip for this kind of lookup is at most a few points
- **The task was far easier than 2023's:** an out-of-place sentence that shares its unusual words with the question. The original effect came from answers hidden among look-alike documents

Why "shares its unusual words" matters

"Turbine K-417" appears once in a novel. Finding it is closer to a text search than to understanding. Real questions rarely match their answer that neatly.

Three things the logs also showed

Missing is not zero

Gemini at minimal used at most 4 output tokens on any call, reasoning included, yet reported no reasoning count at all.

Record a missing field as unknown, not as zero.

Tokenisers differ

For the same documents, Claude Sonnet 5 counted **42% more** input tokens than GPT-5.6 Terra. "64,000 tokens" is a different amount of text to each model.

Compare models on the same text, not the same token count (S1 E2).

Automatic caching

Gemini served 25% of its input from its own cache, because paired prompts share an opening.

Caching changes cost, not accuracy. Shared openings are what get cached.

Reproduce it

Two routes, from the applied-ai-lab repository. Start with the first: it is free.

```
# 1. Rebuild the charts from published data.  
#     No cost, no API key.  
uv run lab analyse \  
    field-notes/s1-e7-lost-in-the-middle/config.yaml  
  
# 2. Rerun against current models,  
#     with your own key.  
uv run lab run \  
    field-notes/s1-e7-lost-in-the-middle/config.yaml \  
    --fresh
```

- Interrupted? Run the same command again: it picks up where it stopped
- Ran it on a newer model and got a different result? Post it in the comments

github.com/BrendanKellyAI/applied-ai-lab

Front-loading a prompt is no longer the fix

For looking up one distinctive fact in up to 64,000 tokens, putting it at the start or end bought nothing for these models. Real documents are harder: look-alike text, several facts, reasoning across the whole.

Ask your team

Does our long-context feature look up one fact, or combine several among look-alikes? Have we tested that shape?

Decision rule

Stop arranging long prompts by position for simple lookups. Test the harder shapes of your own task before trusting a long context.

What would make it fail?

The natural harder tests:

Look-alikes	Several turbine codes in the document, one right answer
Two facts	Questions that need facts from different parts of the document
Longer contexts	Well beyond 64,000 tokens. That length is only a fraction of these models' windows

Which one matches your use case? Seen long context fail in production? Tell me in the comments.

Next: S1 E8 Sampling

How a model picks each token, and why answers vary.

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](https://www.linkedin.com/in/brendan-kelly-irl)
-  **Episode guide** brendankellyai.github.io
-  **Code** github.com/BrendanKellyAI/applied-ai-lab