

Season 1: How LLMs work

# Context windows.

What fits in a model's view, and what happens at the edge

# One budget for everything

- **Shared space:** instructions, history, documents, tool results, and the answer all draw on the same limit
- **Cost:** every token in the window is paid for on every call (S1 E2)
- **Quality:** more text in view is not better attention (S1 E4)
- **Failure:** past the limit, content is rejected or cut

# The window is the model's entire view

A model sees only the tokens sent in this call, up to a fixed limit. Anything outside the window does not exist for it (S1 E1).

## Where it breaks

The limit is counted in the model's own tokens, so how much text fits depends on the tokeniser and the language (S1 E2).

# What fills a window

One call to a typical assistant, drawn to one illustrative budget.



|                      |     |
|----------------------|-----|
| System instructions  | 9%  |
| Tool definitions     | 8%  |
| Conversation history | 27% |
| Retrieved documents  | 30% |
| User message         | 6%  |
| Room for the answer  | 20% |

The answer must fit too: on most models, input and output share one limit

Illustrative proportions. Real ones depend on the application.

# What happens at the edge

**Rejected**

The request is too long, and the API returns an error

**Truncated**

Your framework drops the oldest messages to make room, often without saying so

**Cut off**

The answer hits the output limit. Only a status field says so: "incomplete", reason "max\_output\_tokens"

**Summarised**

Older turns are replaced by a summary, and the detail in them is lost

**Where it breaks**

Only the first of these fails loudly. The other three come back with no error at all.

# An empty answer, billed in full

"Explain how a context window works, in five paragraphs." Three output limits.

| Limit | Status     | Visible text    | Reasoning / output |
|-------|------------|-----------------|--------------------|
| 60    | incomplete | empty           | 60 / 60            |
| 16    | incomplete | empty           | 16 / 16            |
| 8,000 | completed  | five paragraphs | 42 / 401           |

■ The slide's own call: every token spent on hidden reasoning

No text and no error. The caller is billed for 60 output tokens, and only the status field says anything went wrong.

gpt-6-astra, Responses API, 20 September 2026. Hidden reasoning varied: 42 tokens when the limit was generous.

# Trimmed to fit, the instruction vanished

A 50-message conversation opening with "Reply only in French.", trimmed to a 2,000-token budget.

Drop oldest first

**0 of 5**

replies in French

Kept 39 messages, 1,948 tokens

Instruction dropped

Keep the instruction

**5 of 5**

replies in French

Kept 40 messages, 1,953 tokens

Instruction kept

■ Green marks the strategy with fewer French replies

Nearly the same messages, the same budget, no error either way. Every reply from the naive trim came back in English.

gpt-6-astra, 5 runs each, 20 September 2026. Budget counted with o200k\_base (S1 E2), standing in for the model's own tokeniser; a real window is far larger.

# Run it yourself

The cut-off call, with the same client as S1 E1.

```
from openai import OpenAI

client = OpenAI()
response = client.responses.create(
    model="gpt-6-astra",
    input="Explain how a context window works, "
        "in five paragraphs.",
    max_output_tokens=60,
)
print(response.output_text)
print("Status:", response.status)
print("Reason:", response.incomplete_details.reason)
```

■ The first print is empty. The status lines are the only signal

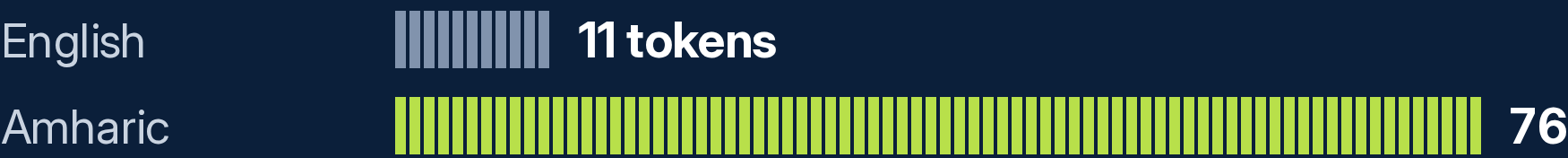
[episodes/s1-e6-context-windows/edge.py](#). The reason is only set when the status is "incomplete".

# The same window holds different amounts of text

Measured in episodes of this season.

## By language: the same sentence

S1 E2



A window holds about a seventh as much Amharic as English.

## By tokeniser: the same documents

S1 E7



Same text, 42% more tokens to fit.

## By era: how much is in view

S1 E5, S1 E7



Even 64,000 tokens is a fraction of current windows.

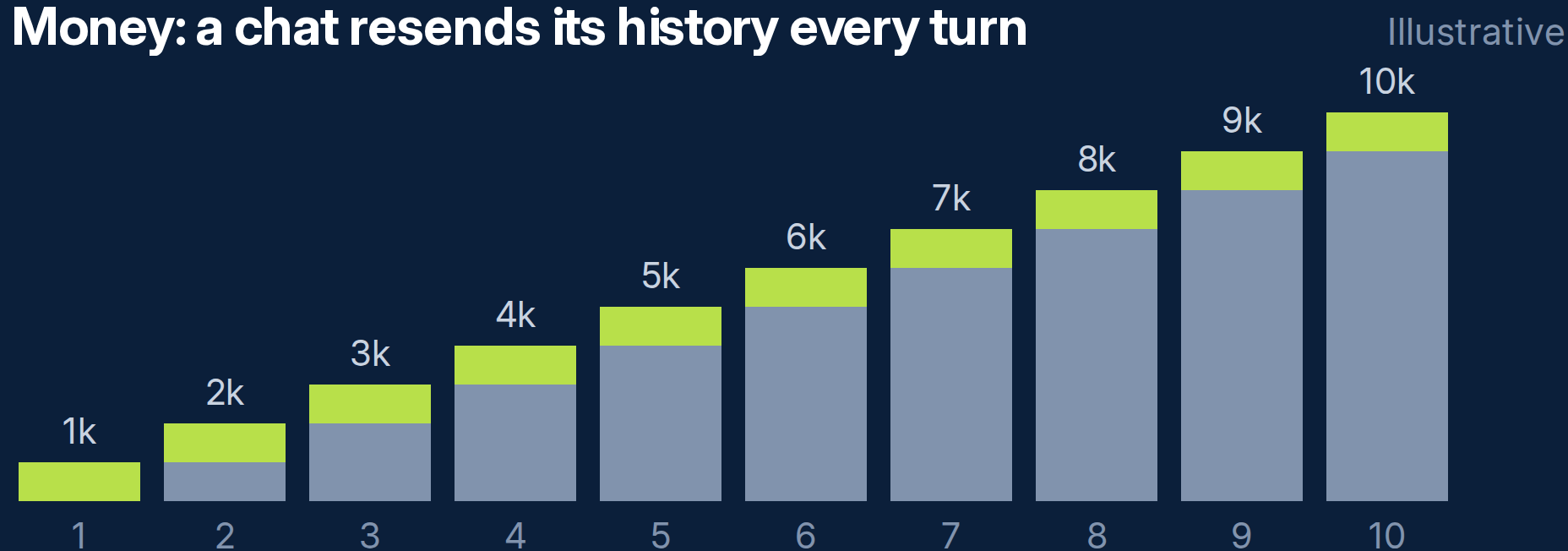
### The consequence

"Fits in the window" is not a property of a document. It depends on the model and the language.

# Filling the window costs time and money

Every token in the window is paid for on every call (S1 E2).

## Money: a chat resends its history every turn



Turn number, with input tokens sent on that call

■ New this turn: 1,000 tokens ■ Resent history

Ten turns bill **55,000** input tokens for **10,000** tokens of conversation.

### Compute

# 100x

the comparisons for 10x the context.  
About 5 billion token pairs at 100,000 tokens (S1 E4).

### Latency

# ~3 s

reading 1,000 prompt tokens before the first word appears: GPT-2 on a laptop, 2.9 ms per token (S1 E5).

# A prompt that fits is not a prompt that works

Fitting is a size check. Working is a quality check.

0 tokens

Not to scale

Advertised limit

## Easy lookup

One distinctive fact, hidden anywhere in the text

**Found at every position**

**Held** up to 64,000 tokens, the largest size S1 E7 tested.

## Hard tasks

Look-alike passages, several facts, reasoning across the whole document

**Not tested here**

**Widely reported** to fall well before the stated limit.

### And when it does not fit

Truncation drops content with no error to catch. On slide 7, the naive trim lost the instruction: 0 of 5 replies in French.

# Managing the window

Five habits, in the order a call runs.

## BEFORE YOU SEND

### 1 Count

Use the model's own tokeniser or the provider's counting endpoint (S1 E2).

```
tokens = count_tokens(model, messages)
```

### 2 Reserve the answer

Set the output limit first. The input gets what is left.

```
input_budget = window - max_output_tokens
```

### 3 Pin, then drop

Keep the instructions. Drop old turns or retrieved passages first.

```
trim(messages, budget, pinned=[system])
```

## AFTER IT RETURNS

### 4 Check the status

An empty answer arrives with no error.

```
if response.status == "incomplete": ...
```

### 5 Log the edge

Token counts, anything dropped, and the status, on every call.

```
log(tokens_in, tokens_out, dropped, status)
```

# What breaks in production

Read it as a diagnosis: the symptom, then the usual cause.

YOU SEE

USUALLY BECAUSE

|                                            |                                                               |
|--------------------------------------------|---------------------------------------------------------------|
| <b>Blank answers, no error</b>             | → The output limit was hit. Only the status field says so.    |
| <b>Long chats forget the rules</b>         | → Truncation dropped the instruction to make room.            |
| <b>Cost climbs with every turn</b>         | → The whole history is resent on every call.                  |
| <b>A big bill for short answers</b>        | → Whole documents pasted in, when the question needed a page. |
| <b>Passed testing, fails in production</b> | → Tested on a page, run on a hundred.                         |

# A bigger window is a budget, not a solution

A large window lets a team skip deciding what the model needs, and then pay for that on every call.

## Ask your team

What is our largest real input, and have we tested quality, cost, and speed at that size?

Decision rule

**Treat the context window as a budget: count what you send, reserve room for the answer, and test at your real size.**

## Next: S1 E7 Lost in the middle

Does a fact's position in a long document still decide whether a model finds it? Tested.

Where this fits

# The series map

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

☐ Leadership season

|                                                                                                          |                                                                                                         |
|----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
|  <b>Follow</b>        | <a href="https://www.linkedin.com/in/brendan-kelly-irl">linkedin.com/in/brendan-kelly-irl</a>           |
|  <b>Episode guide</b> | <a href="https://brendankellyai.github.io">brendankellyai.github.io</a>                                 |
|  <b>Code</b>          | <a href="https://github.com/BrendanKellyAI/applied-ai-lab">github.com/BrendanKellyAI/applied-ai-lab</a> |