

Season 1: How LLMs work

Sampling.

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

Why the same prompt gives different answers

- **Variation:** identical prompts can return different answers (S1 E1)
- **Testing:** one good run proves little about the next one
- **Control:** a few settings trade consistency against variety

After this episode, you can

- Explain where the variation comes from
- Read temperature, top-k, top-p, and seed
- Predict how much a model will vary, and why a chat model varies less than you expect
- Test a feature whose answers vary

The model scores. A weighted draw chooses.

Every step produces a probability for every token (S1 E1). Sampling picks one, in proportion to those probabilities.

1. Score

The model gives every token in its vocabulary a score



2. Convert

Scores become probabilities that add up to 100%



3. Draw

One token is drawn. Likely tokens win more often, not always

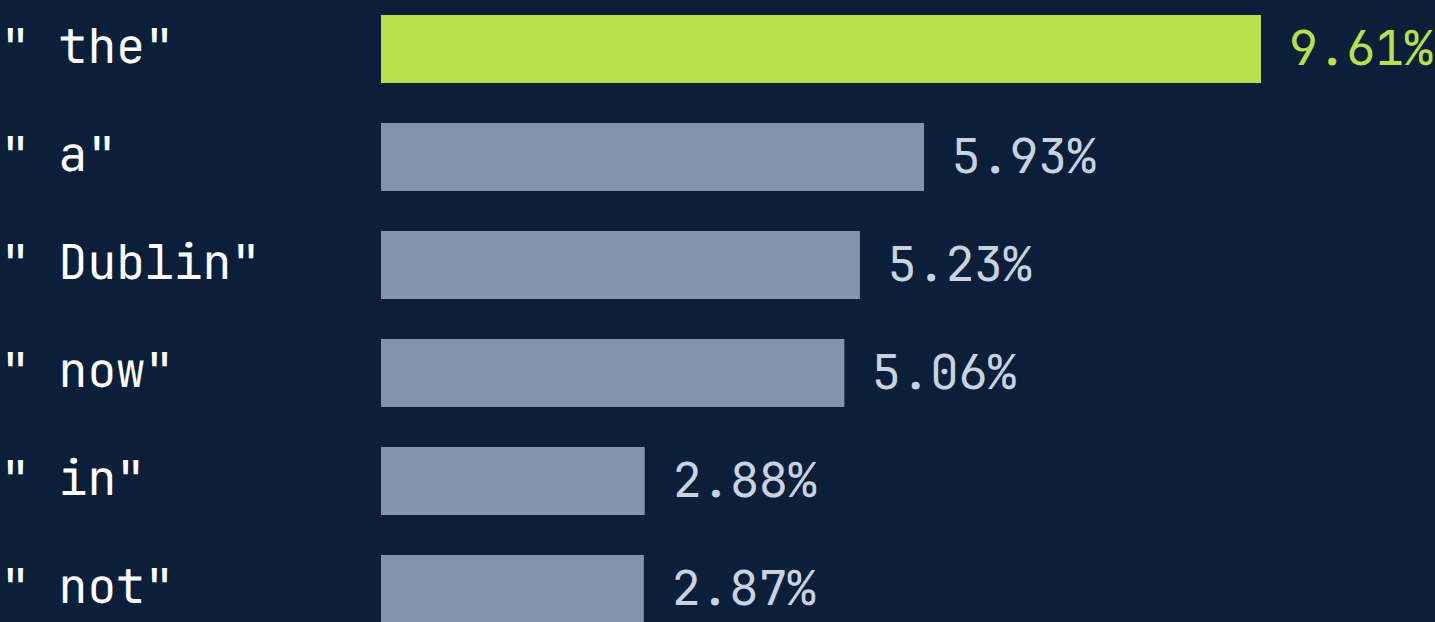
Repeat for every token in the answer. The settings on the next slides act on steps 2 and 3, never on the model's knowledge.

Where it breaks

Turning randomness down is not a guarantee of identical output. Providers' serving systems can still vary from run to run.

Greedy picks the top. Sampling rolls the dice.

GPT-2 small, next token after "The capital of Ireland is"

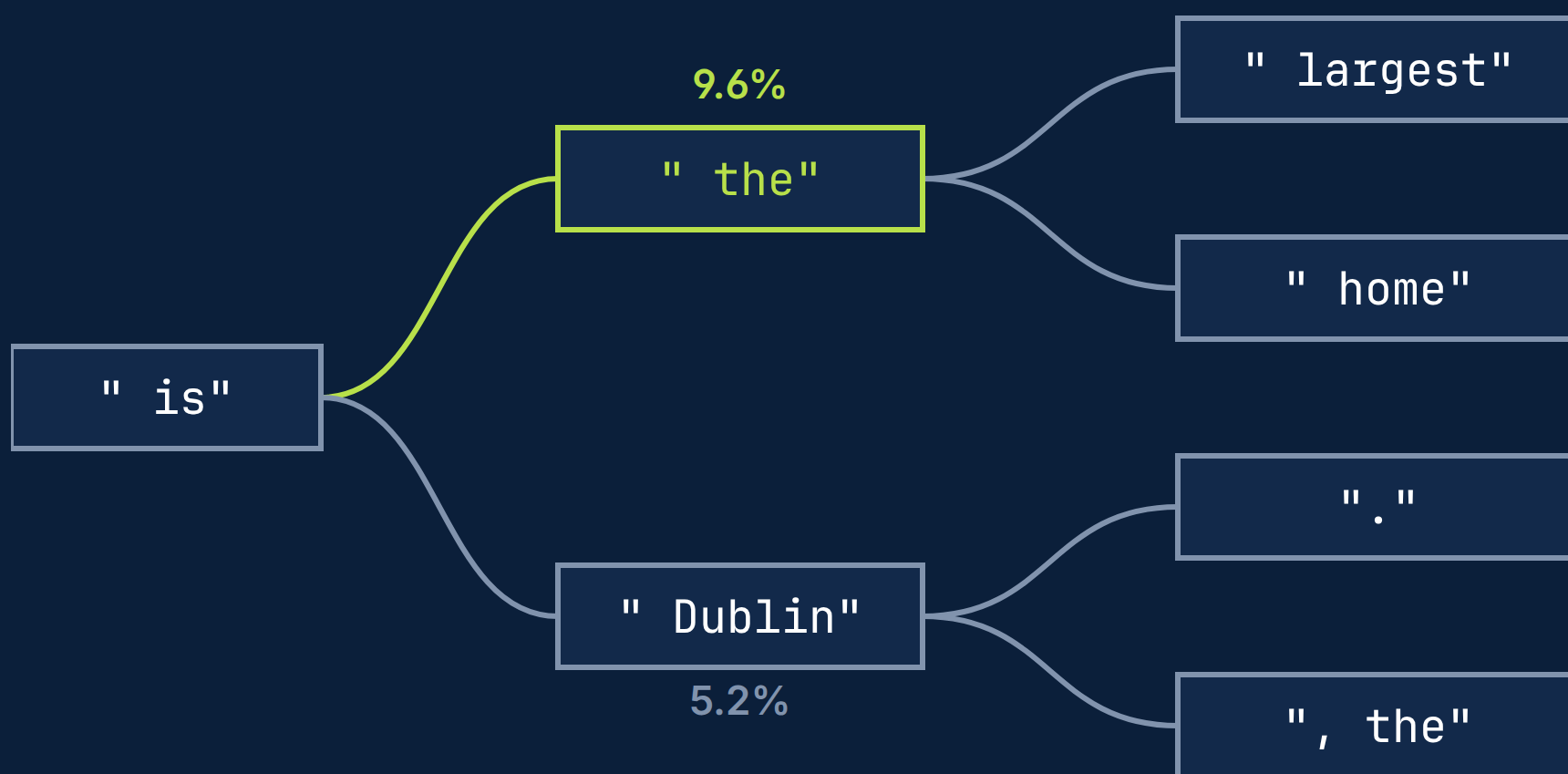


Greedy	Always " the", the single most likely token
Sampling	" the" about 1 time in 10, " Dublin" about 1 in 19, and thousands of other tokens the rest of the time

The top 10 tokens hold only 40% of the probability. S1 E1's 94% for " Dublin" was illustrative; a small 2019 model is far less sure.

One different token, a different answer

Each token is chosen given everything before it. An early change sends the rest of the answer down another path.

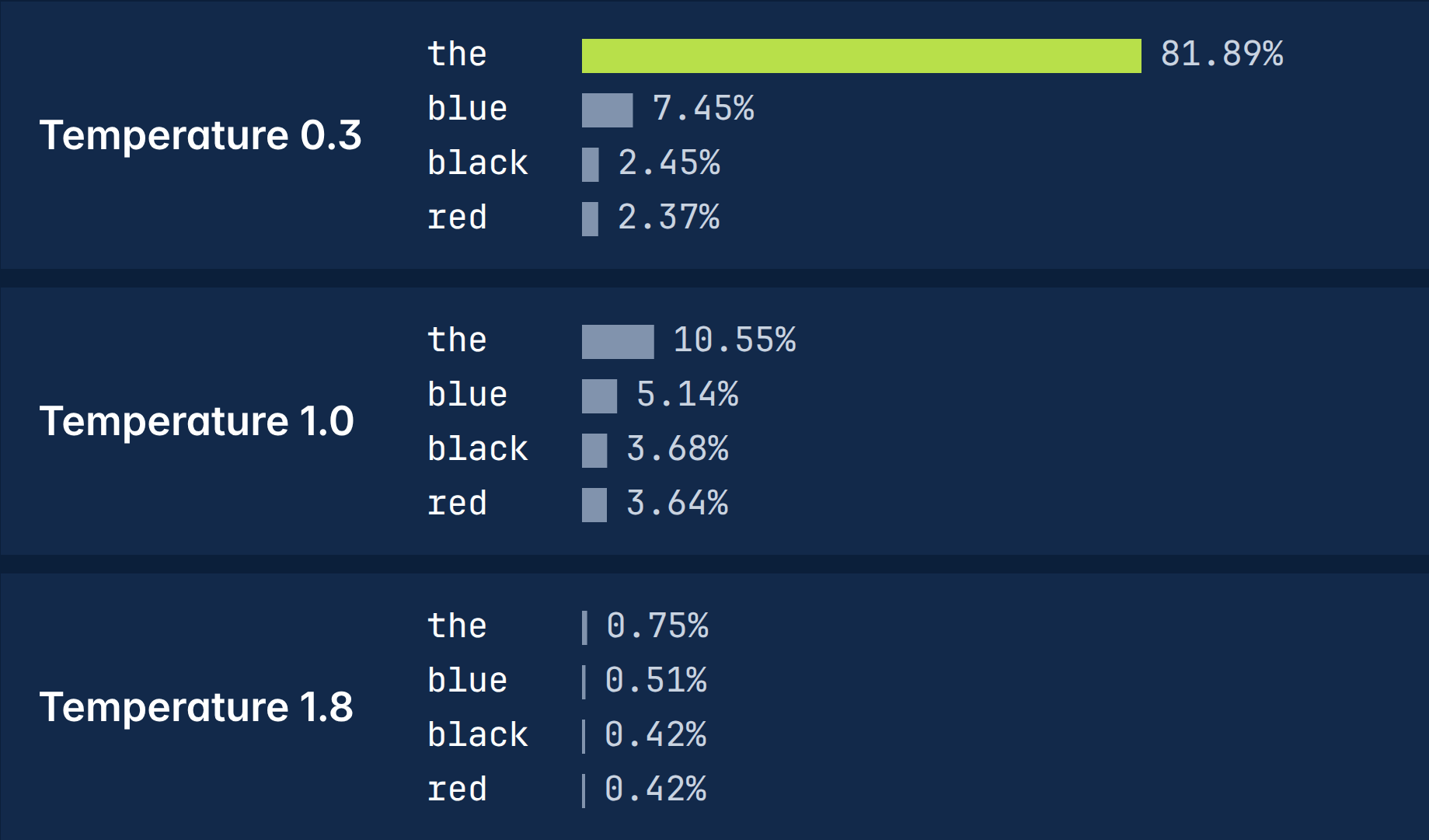


■ The most likely next token. The branch below starts from a 5% draw
First step from GPT-2 small, as on the previous slide. Later tokens illustrative.

Temperature reshapes the odds

Temperature rescales the scores before they become probabilities. Below 1 it sharpens the odds, above 1 it flattens them, and near 0 it behaves like greedy.

Example: GPT-2 small, after "My favourite colour is". Top 4 of 50,257 tokens.

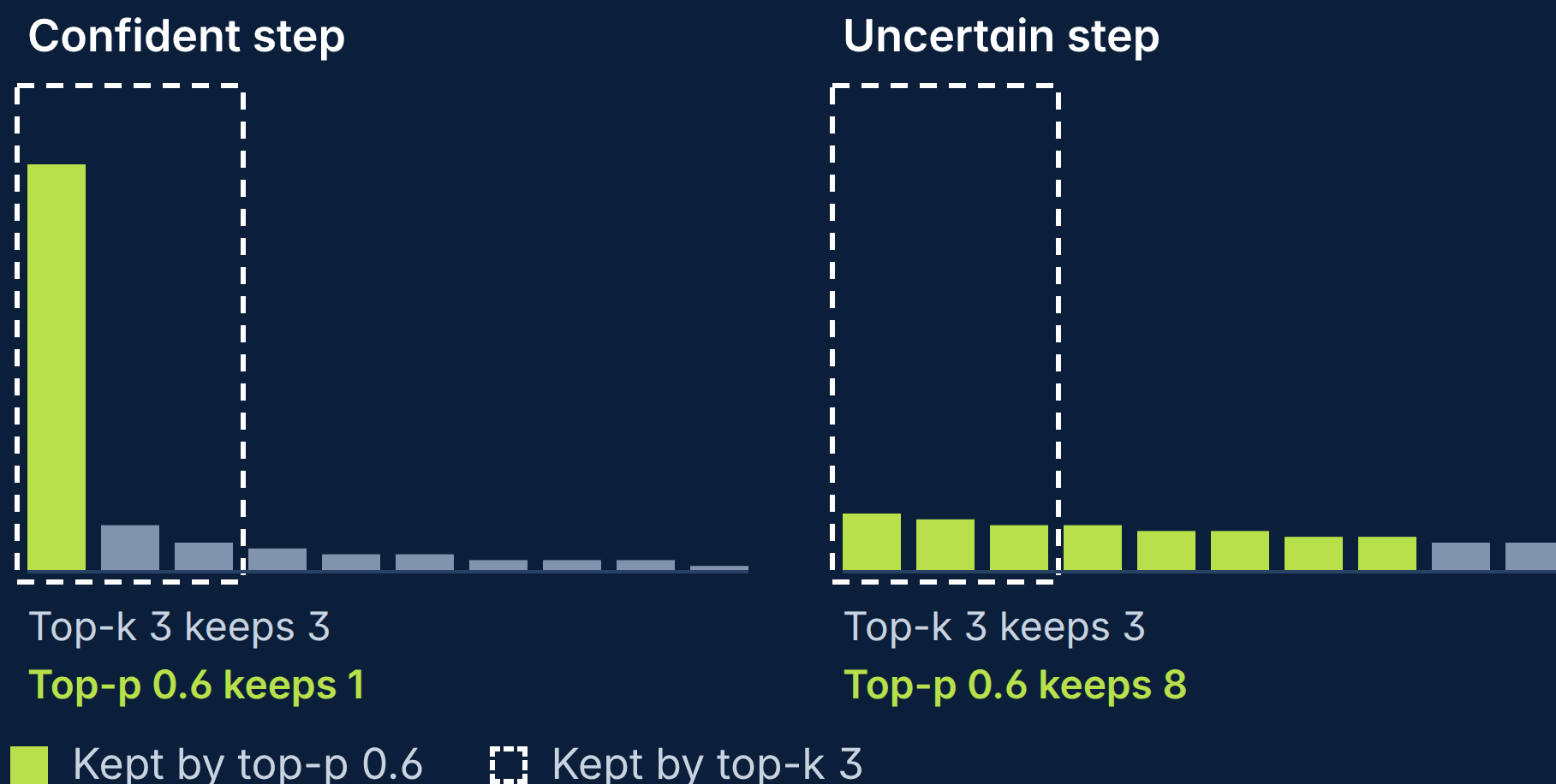


■ At 0.3, " the" takes 82% of the draws. At 1.8, no token reaches 1%

Computed exactly from the model's scores. The ranking never changes; only how much the favourite wins by.

Top-k and top-p cut the tail

Both remove unlikely tokens before the draw. **Top-k** keeps a fixed number. **Top-p** keeps the fewest tokens whose probabilities add up to p .



Real example

On the uncertain colour prompt, top-p 0.6 keeps 63 tokens. Top-p adapts to how sure the model is; top-k does not.

Bars illustrative.

Cheat sheet: four settings, one line each

Temperature	Sharpens the odds below 1, flattens them above 1. Near 0, it behaves like greedy
--------------------	--

Top-k	Draw only from the k most likely tokens
--------------	---

Top-p	Draw only from the smallest set whose probabilities add up to p. At $p = 0.6$, GPT-2 keeps 63 tokens for the colour prompt
--------------	---

Seed	Fixes the random draw, where a provider offers it, on a best-effort basis
-------------	---

Not every provider or model exposes every setting.

Save this

All four act on the draw. None of them changes what the model knows, so none of them can make an answer true.

Predict before you swipe

A current chat model, asked 20 times: "Suggest a name for a coffee shop in Dublin. Reply with the name only." Same prompt, default temperature of 1.0. How many different names come back?

A 1: the same name every time

B 2 to 5 names

C 10 to 15 names

D 20: a new name every call

Hold your answer

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

Answer B: 20 identical calls, 2 coffee shops

gpt-6-astra, the same prompt 20 times, at its defaults.

Prompt	Distinct	Answers
Capital of Ireland	1	Dublin (20)
Coffee shop name	2	The Liffey Grind (17), Liffey & Latte (3)
GPT-2 small, colour	20	20 different continuations, at every temperature

■ The open-ended prompt: hundreds of valid names exist, yet only two came back

Variation depends on the question and the model, not only the settings. One run of each, 20 September 2026.

gpt-6-astra rejects a temperature setting ("Unsupported parameter"); every response ran at 1.0. GPT-2 continued for 10 tokens, seeds 0 to 19.

Same temperature, very different spread

Temperature 1.0 means "sample from the model's own odds". How spread out those odds are depends on how the model was trained.

Base model

GPT-2 was trained only to continue internet text. Its odds are wide, so 20 draws gave 20 different continuations

Chat model

Chat models are tuned further to give the answers people prefer. That tuning tends to pile probability onto a few favourites, so even at 1.0 the same names keep winning

What to do with this

A temperature number does not tell you how much a model varies. Measure the spread on your own prompts.

The two rows used different prompts, so this is a contrast, not a controlled comparison.

Run it yourself

Twenty identical calls, counted.

```
from collections import Counter
from openai import OpenAI

client = OpenAI()
prompt = ("Suggest a name for a coffee shop in Dublin. "
          "Reply with the name only.")
responses = [
    client.responses.create(
        model="gpt-6-astra", input=prompt,
        max_output_tokens=4000)
    for _ in range(20)
]
counts = Counter(r.output_text.strip() for r in responses)
for answer, count in counts.most_common():
    print(f"{count:2d} {answer}")
```

- A generous output limit, so hidden reasoning cannot empty the answer (S1 E6)
- Try it: swap in one of your own prompts. How many distinct answers do you get?

[episodes/s1-e8-sampling/sampling.py](#). GPT-2's probabilities come from [local.py](#), which needs no key.

Sampling settings do not make answers true

- **Low is not deterministic:** widely reported, and not testable here: gpt-6-astra rejects a temperature setting
- **Low is not correct:** it picks the most plausible token, and plausible can be wrong (S1 E1)
- **High invents:** flattened odds give unlikely tokens a real chance, including wrong facts
- **One run is an anecdote:** a test that passed once can fail on the next call

The capital of Ireland, at temperature 1.8

Temperature flattens every prompt the same way. At 1.8, "Dublin" would lose most of its edge over thousands of other tokens, wrong ones included.

What breaks in production, and the fix

Demo luck

The answer that impressed in the meeting does not come back.

Run each test case several times and report how often it passes.

Flaky tests

Checks that compare exact text fail at random.

Check what the answer must contain, and track a rate: "passed 18 of 20".

Creative settings on facts

High temperature on extraction puts invented values in the output.

Set it by task: low for extraction and classification, higher for drafting and ideas.

Unrecorded settings

Nobody knows what the model ran with when it went wrong.

Store the settings with every call, and check the defaults: gpt-6-astra rejects temperature and runs at 1.0.

Variation is a setting, and a risk to measure

Every answer from a model is one draw from many possible answers. Judging a feature on one draw is judging it on luck.

Ask your team

Should this feature give the same answer every time, and how often does it actually differ?

Decision rule

Set sampling for the task, and judge a model by its rate across many runs, never by one answer.

What temperature do you run in production, and how did you choose it? Tell me in the comments.

Next: S1 E9 Hallucination

Why fluent answers can be confidently wrong.

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