THE LAB — DAY 08

Machines That Read

How language models read, write, and sometimes make things up

Day 08 · Week 2 · Unlocks Language Lab

1 Mission briefing

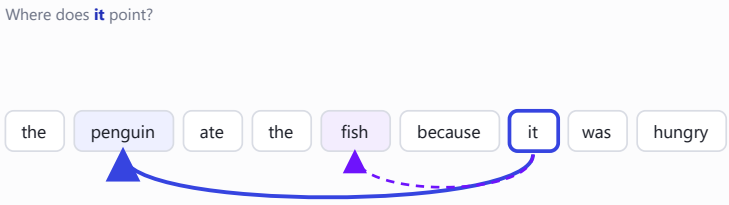
Your models can see. Now make one **read**. Autoregressive models such as GPT-2 generate text by predicting the next token from context. Their fluent output does not guarantee factual accuracy. Today you crack open GPT-2 and friends to watch tokens, embeddings, attention, and temperature at work — and catch a model sounding perfectly confident while being perfectly wrong. *Previously:* you unlocked RPS Arena, transfer learning tested against your own hands.

- ☐ Break text into **tokens** (and see why an LLM miscounts letters)
- ☐ Watch **next-token prediction**, the whole secret
- ☐ Peek at **attention**: which words look at which
- ☐ Turn words into **embeddings** — meaning as geometry
- ☐ Turn the **temperature** knob from robotic to unhinged
- ☐ Catch a **hallucination**: plausible ≠ true

2 Field guide the concepts

Tokens, not letters. A model reads **tokens** — common chunks of text, like puzzle pieces. "strawberry" might be two or three pieces, which is exactly why asking an LLM to count the letters in a word is weirdly hard for it.

Embeddings are meaning as geometry. Each token becomes a long arrow of numbers (768 of them for GPT-2). Similar meanings point similar directions — so much so that `king − man + woman ≈ queen`. And the entire engine is **next-token prediction**: given the text so far, score every possible next token, pick one, append it, repeat. Essays, code, and poems are all this one move on a loop, with **temperature** setting how boldly it picks.



Reading "...the penguin ate the fish because **it** was hungry", this schematic assigns attention from **it** to **penguin** and **fish**. Real weights vary across layers and heads; they do not prove the pronoun's reference.

Plausible ≠ true. A fluent, confident sentence can still be flat wrong — a **hallucination**. The model is trained to sound plausible, not to be correct. Always verify.

> **Matrix Watch —** attention is matrix multiplication too: every token's query vector times every token's key vector ($Q \cdot K^T$) is one big matmul — the very operation HPC engineers make fast.

A precise reading

GPT-2 predicts the next token, which may be a word fragment. Encoder models such as DistilBERT can use both sides of a masked token. Tokenization can make letter counting harder but does not make it impossible. An attention heatmap shows model weights, not proof of grammatical reference or a complete explanation of a prediction. A lower temperature changes sampling, not factual reliability.

3 Lab missions check each off in the notebook

1 Find a word that shatters ☐
Tokenize words until one splits into several pieces.
Expect: a familiar word broken into surprising sub-word tokens.

2 The strawberry problem 🍓 ☐
Ask a model to count a letter in a word, then check by hand.
Expect: it miscount — because it sees tokens, not letters.

3 Put your own words on the map ☐
Embed a few words and compare their vectors.
Expect: related words score as more similar than unrelated ones.

4 Predict from your own prompt ☐
Feed a story starter to a `text-generation` pipeline.
Expect: GPT-2 continue your sentence, token by token.

5 Fool the mood detector 😏 ☐
Craft a sentence that flips the sentiment classifier.
Expect: a sarcastic line the model reads the 'wrong' way.

6 Two temperatures 🌡️ ☐
Generate the same prompt at temperature 0.2 and 1.3.
Expect: safe-but-boring vs wild-but-creative text.

7 Follow the pronoun ☐
Change the sentence and compare the DistilBERT attention heatmap.
Expect: attention weights that may shift; no single head proves the pronoun's reference.

4 Cheat sheet transformers pipelines

`tok = AutoTokenizer.from_pretrained("gpt2")`	Load GPT-2's tokenizer.
`tok.tokenize("strawberry")`	See the puzzle pieces a model reads.
`pipeline("text-classification", model= ... )`	One-line sentiment / mood classifier.
`pipeline("text-generation", model="gpt2")`	Load GPT-2 as a text generator.
`gen(p, max_new_tokens=60, do_sample=True)`	Generate ~60 tokens with sampling on.
`temperature=0.9, pad_token_id=50256`	Creativity knob · GPT-2's end-of-text id.
`pipeline("fill-mask", model= ... )`	Predict a masked-out word.
`s.replace("___", tok.mask_token)`	Swap your blank for the token the model expects.
`pipeline("zero-shot-classification", ... )`	Sort text into labels you invent at runtime.

5 🔓 Unlock your Studio module

Language Lab

Sign your work and set a default temperature; the transformer models load from the Hugging Face cache, so the unlock cell only writes:

`ai_studio/models/language_lab.json`

Four language superpowers — mood, story, fill-the-blank, sorting hat — go live:

```
$ python ai_studio/app.py
```

6 Mini-glossary

token
A chunk of text (word or word-piece) the model actually reads.

embedding
A token turned into a vector; meaning becomes direction.

transformer
The attention-based architecture behind modern language models.

attention
The mechanism letting each token weigh the others for context.

temperature
The knob controlling randomness in generation.

LLM
Large language model — a giant next-token predictor.

hallucination
A fluent, confident output that simply is not true.

7 Show & tell

1. Show a word your tokenizer split weirdly. Why does that make letter-counting hard?
2. Same prompt at low vs high temperature — which felt more useful, and for what?
3. Show a confident hallucination you caught. How would you fact-check a model in real life?

© 2026 Dr. Abdelghafour HALIMI · ahalimi.com · All rights reserved