上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 27: Large Language Models

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

https://taohuang.info/cs3317

https://oc.sjtu.edu.cn/courses/89538

AI tools assisted in generating some figures in these slides. All such content has been reviewed, and the instructor is responsible for its accuracy.

Where We Left Off

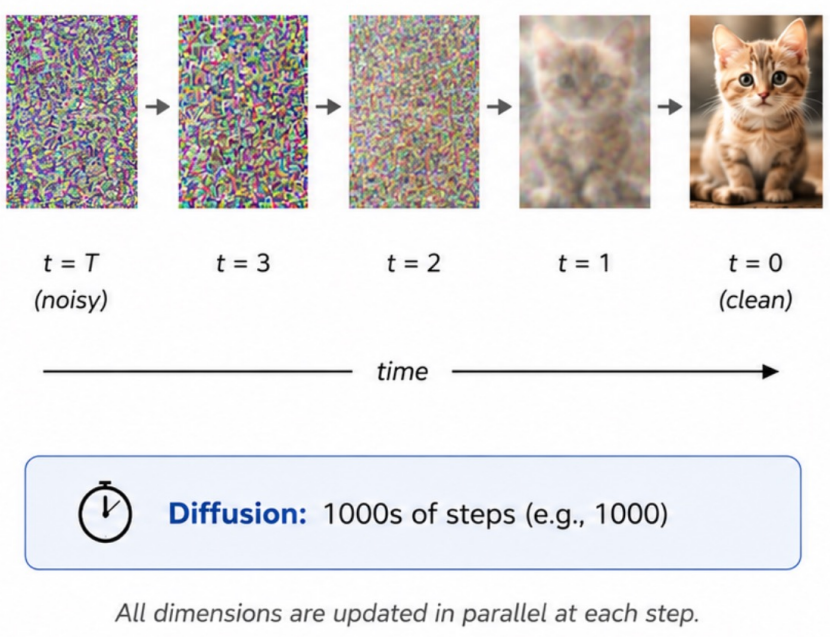
Image generation (continuous):

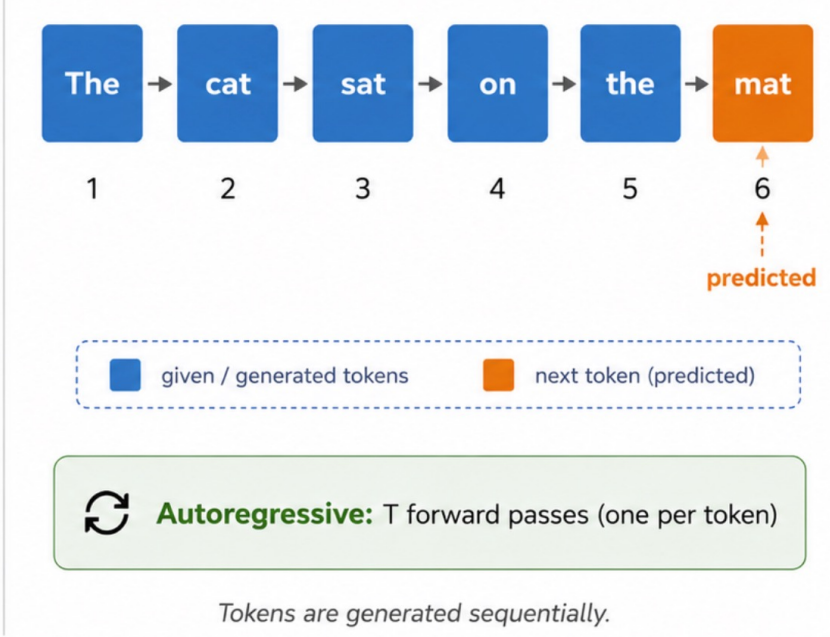
- continuous pixels, no canonical likelihood
- sharpness ↔️ likelihood trade-off
- **diffusion broke the trilemma**

Text generation (discrete):

- discrete tokens
- canonical likelihood (cross-entropy)
- **autoregressive wins by default —**

so why was LLM progress so hard?

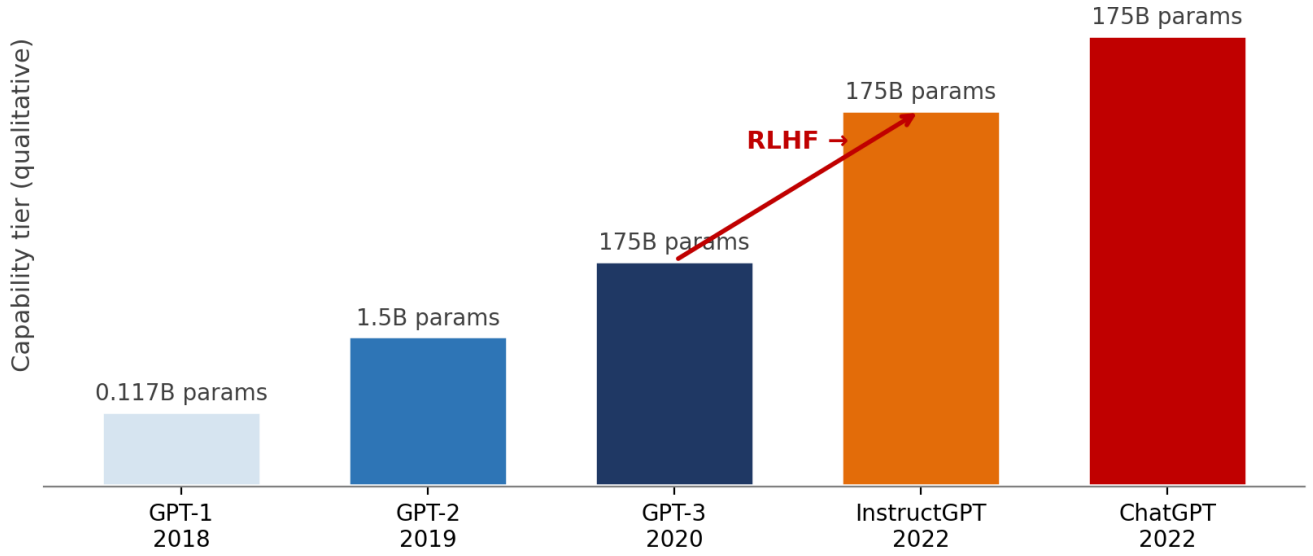




The Real Question

- Next-token prediction: Shannon, 1948.
- Transformer: 2017.
- GPT-1 (117M): 2018 — fine, but unremarkable.
- ChatGPT: 2022 — changed everything.

Architecture barely changed in those four years. What did?



Objectives

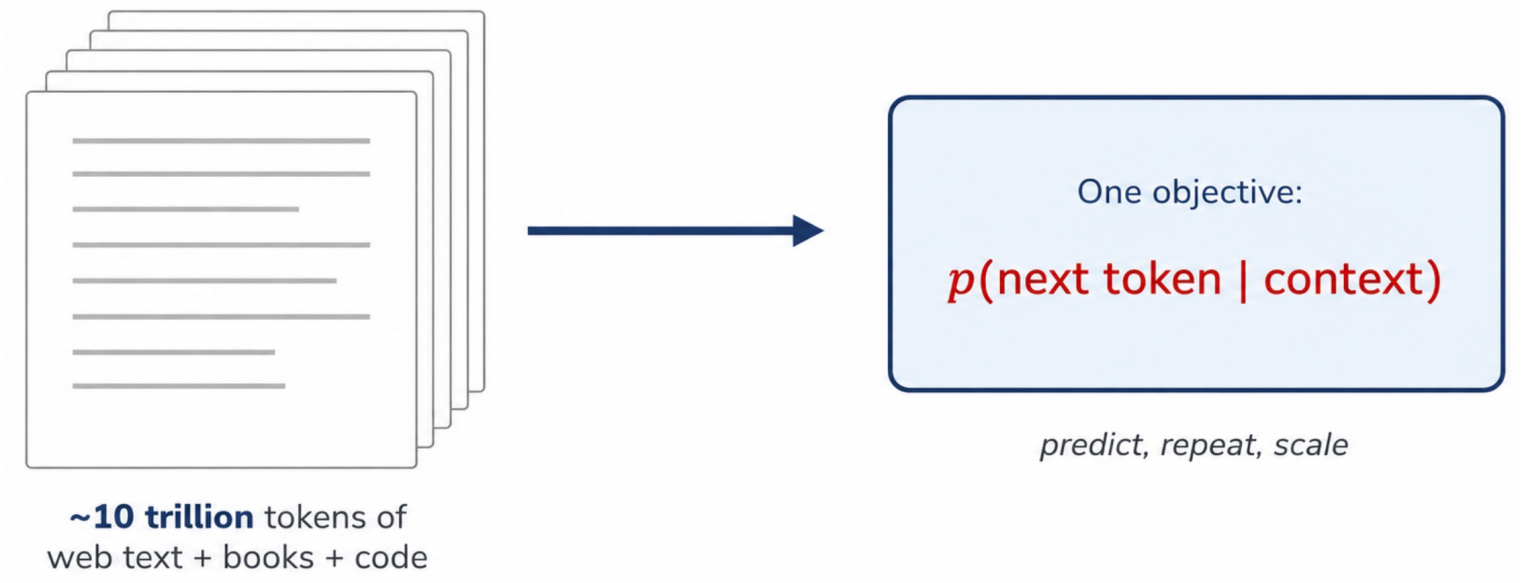
By the end of this lecture, you should be able to:

- Explain why cross-entropy has a fundamental lower bound (entropy of language).
- Derive Chinchilla's compute-optimal N / D ratio.
- Distinguish pretraining, SFT, and RLHF — what each is for.
- Sketch the PPO objective and explain why the KL constraint is essential.
- Evaluate whether LLMs “understand” — and what evidence either side has.

1. Pretraining: The Autoregressive Engine

The Setup

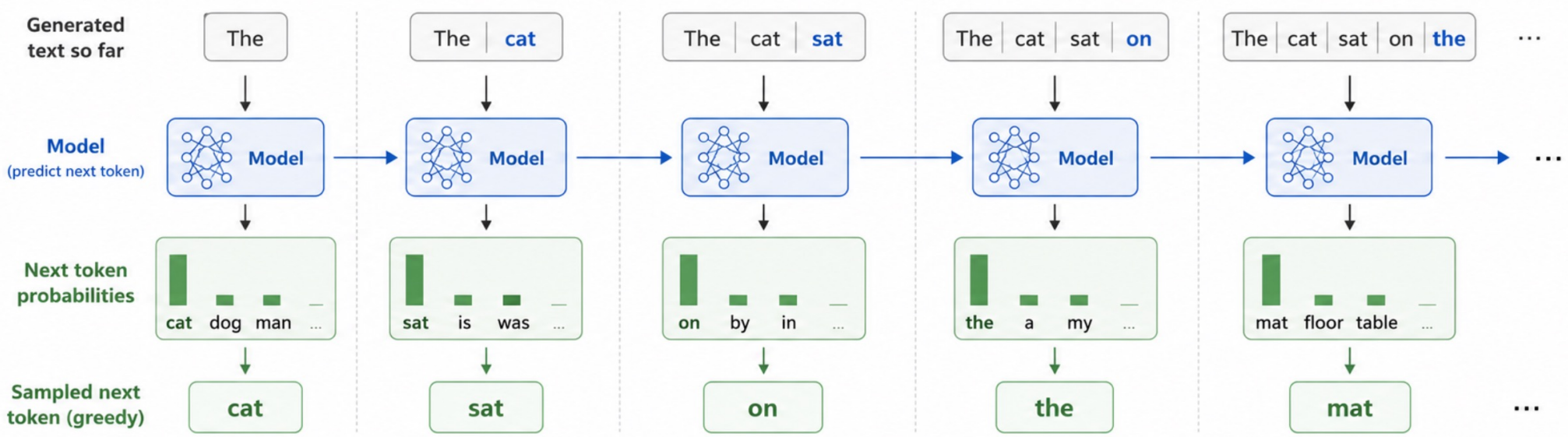
- **Situation:** Terabytes of web text. We want a generative model.
- **Complication:** Text is discrete. No Gaussian noise. No backprop through samples.
- **Question:** *What's the simplest objective that turns text into a generative model?*



Autoregressive Factorization

- Any joint distribution over a sequence factorizes exactly:

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t \mid x_{<t})$$



Joint probability factorization (chain rule):

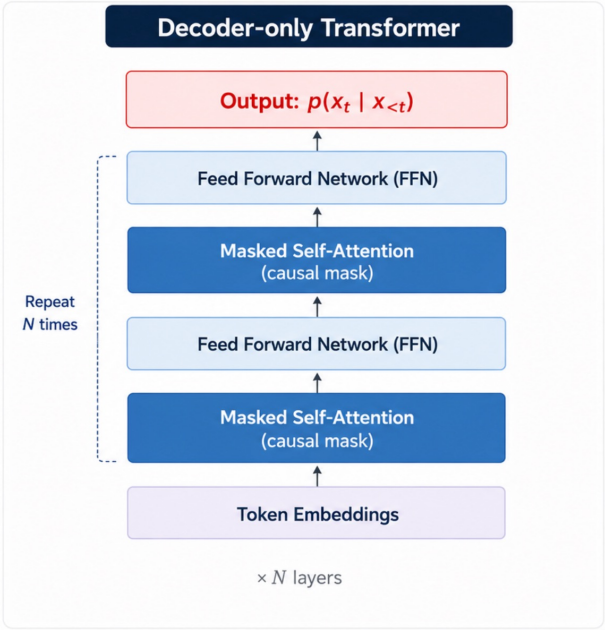
$$p(\text{The, cat, sat, on, the, mat}) = p(\text{The}) \cdot p(\text{cat} \mid \text{The}) \cdot p(\text{sat} \mid \text{The cat}) \cdot p(\text{on} \mid \text{The cat sat}) \cdot p(\text{the} \mid \text{The cat sat on}) \cdot p(\text{mat} \mid \text{The cat sat on the})$$

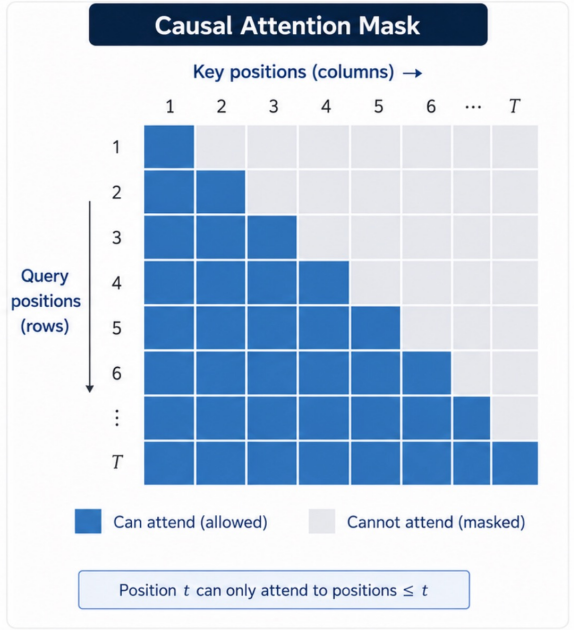
Training Objective: Cross-Entropy

Minimize the negative log-likelihood over the corpus:

$$\mathcal{L}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\log p_\theta(x_t \mid x_{<t})$$

- Standard cross-entropy, summed over positions.
- One forward pass computes all T loss terms in parallel.
- **Trained on ~10^{12}–10^{13} tokens from the web.**





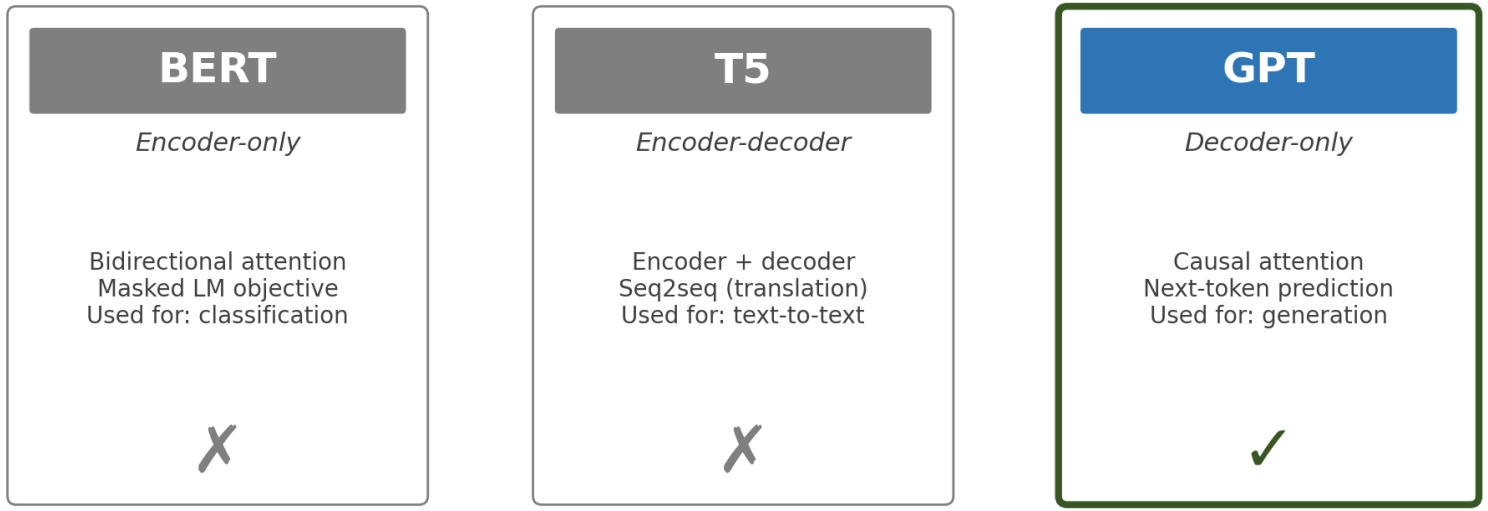
Why Decoder-Only?

Frontier LLMs use decoder-only.

Constraint: Next-token prediction needs no separate encoder — the prefix is the encoding.

Result: Simpler architecture; every position is a target; unifies understanding + generation.

Encoder-decoder was an artifact of supervised translation. Pretraining dissolves it.



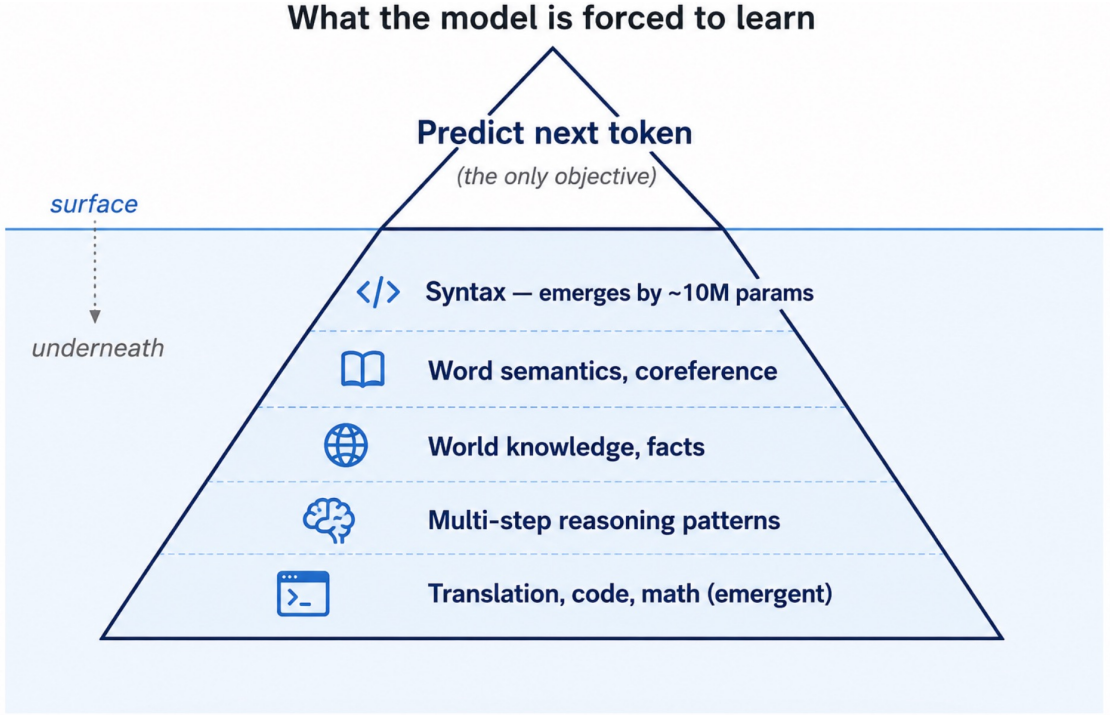
What the Model Actually Learns

Scaling reduces the KL term — the gap to the true distribution.

What the model is forced to learn:

- Syntax (free, by ~10M params)
- Word semantics, coreference
- Facts (the heavy lifting)
- Reasoning patterns (debated)

Pretraining is information compression.
Compression requires structure.

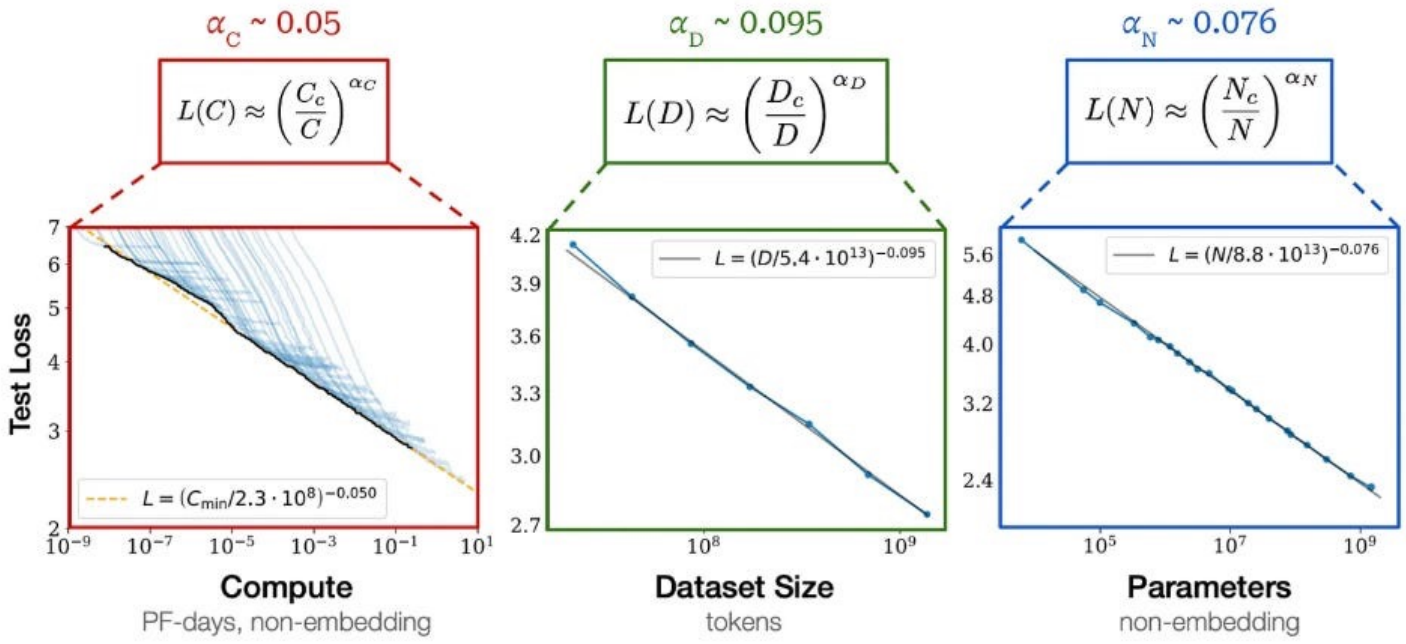


2. Scaling Laws: How Big, How Much Data?

The Empirical Surprise (Kaplan 2020)

- Knobs: N (params), D (data), C (compute).
- Discovery: loss falls as a smooth power law in each, over many orders of magnitude.

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}, \qquad \alpha_N \approx 0.076$$

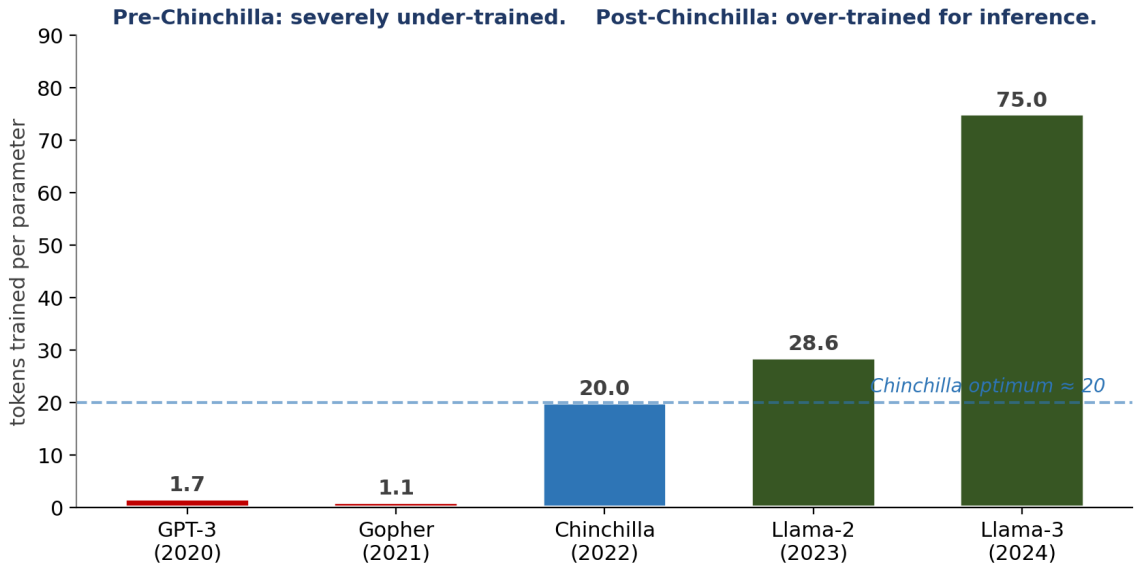


Kaplan's Recommendation (and Its Bug)

- Kaplan et al. [1]:
  model size should grow much faster than data.
- Shaped GPT-3 (175B params, ~300B tokens) and 2 years of frontier models.

GPT-3 was severely under-trained — Kaplan's framework had a bug.

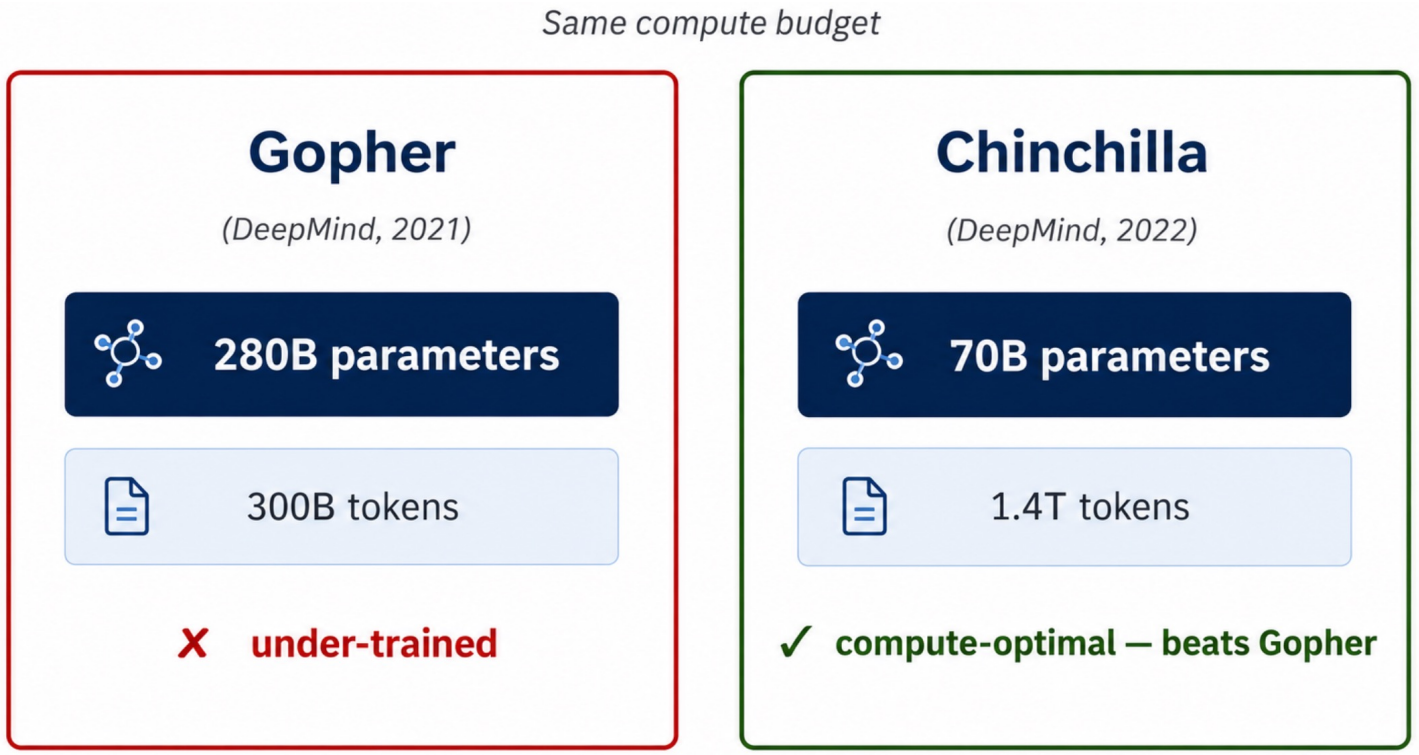
The bug: LR schedule fixed incorrectly across scales.



[1] Kaplan et al., "Scaling Laws for Neural Language Models," 2020.

The Chinchilla Correction

- DeepMind re-ran with proper LR scheduling per model [2].
- Result: N and D should grow at equal rates. Optimum: ~20 tokens / param.



4× smaller, 4× more data, same compute, better.

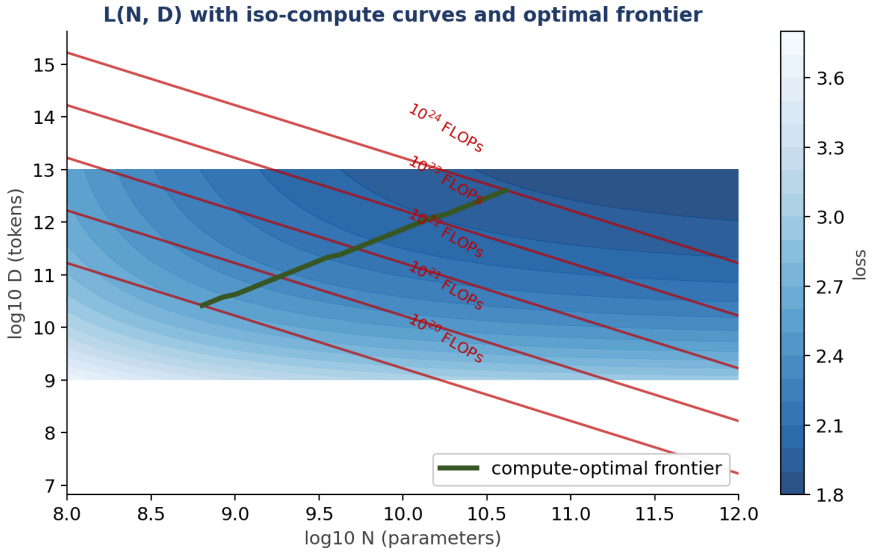
[2] Hoffmann et al., “Training Compute-Optimal Large Language Models" (Chinchilla), NeurIPS 2022.

Compute-Optimal Derivation (Setup)

- Total training FLOPs: $C \approx 6\,N\,D$
- Empirical loss (Hoffmann fit):

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

with $\alpha \approx \beta \approx 0.34$, $E \approx 1.69$ (= the entropy floor).



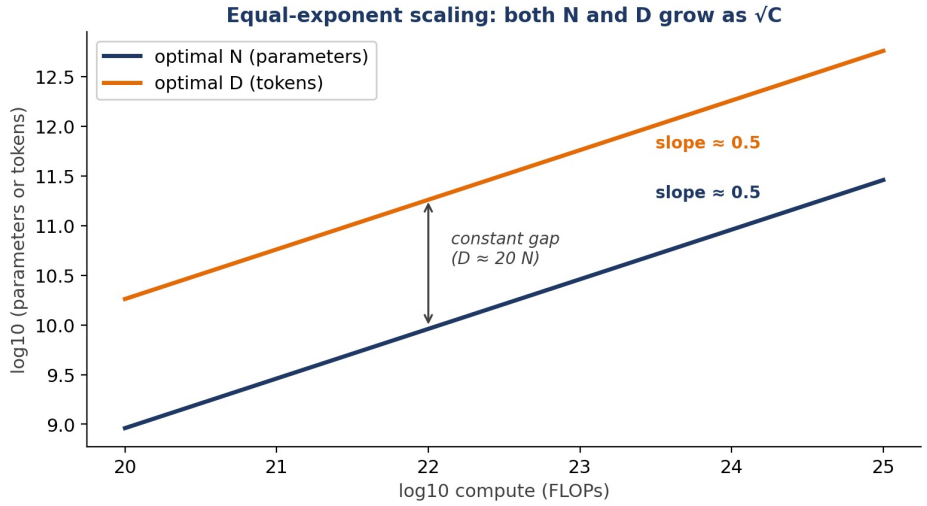
Compute-Optimal Derivation (Result)

Lagrangian: minimize L(N, D) subject to 6 N D = C.

$$\frac{\partial L/\partial N}{\partial L/\partial D} = \frac{N}{D} \quad \Longrightarrow \quad N_{\text{opt}} \propto C^{0.5}, \quad D_{\text{opt}} \propto C^{0.5}$$

Equal exponents. Rule of thumb: D_opt ≈ 20 N_opt.

Want 4× model size? You need 4× data and 16× compute.



Think: Plan a Training Run

- *Compute budget = 10^{23} FLOPs. Per Chinchilla, what N and D?*

- Set up: $C = 6\,N\,D$, $D = 20\,N$ → $N = \sqrt{(C / 120)}$.

- N_opt ≈ 29 B parameters.
- D_opt ≈ 580 B tokens.

- *Compare: Llama-2 70B = 2T tokens (~28 / param).*

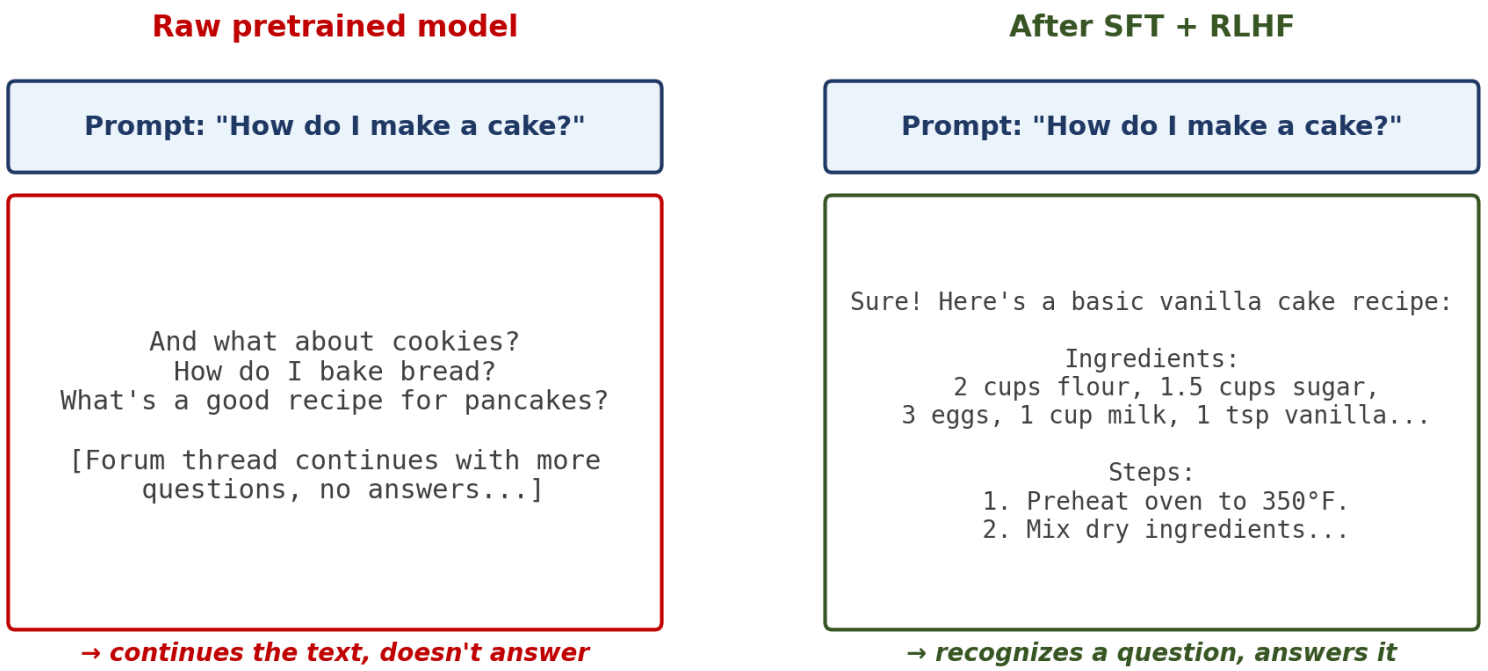
3. Alignment: From Pretrained to Useful

The Alignment Problem

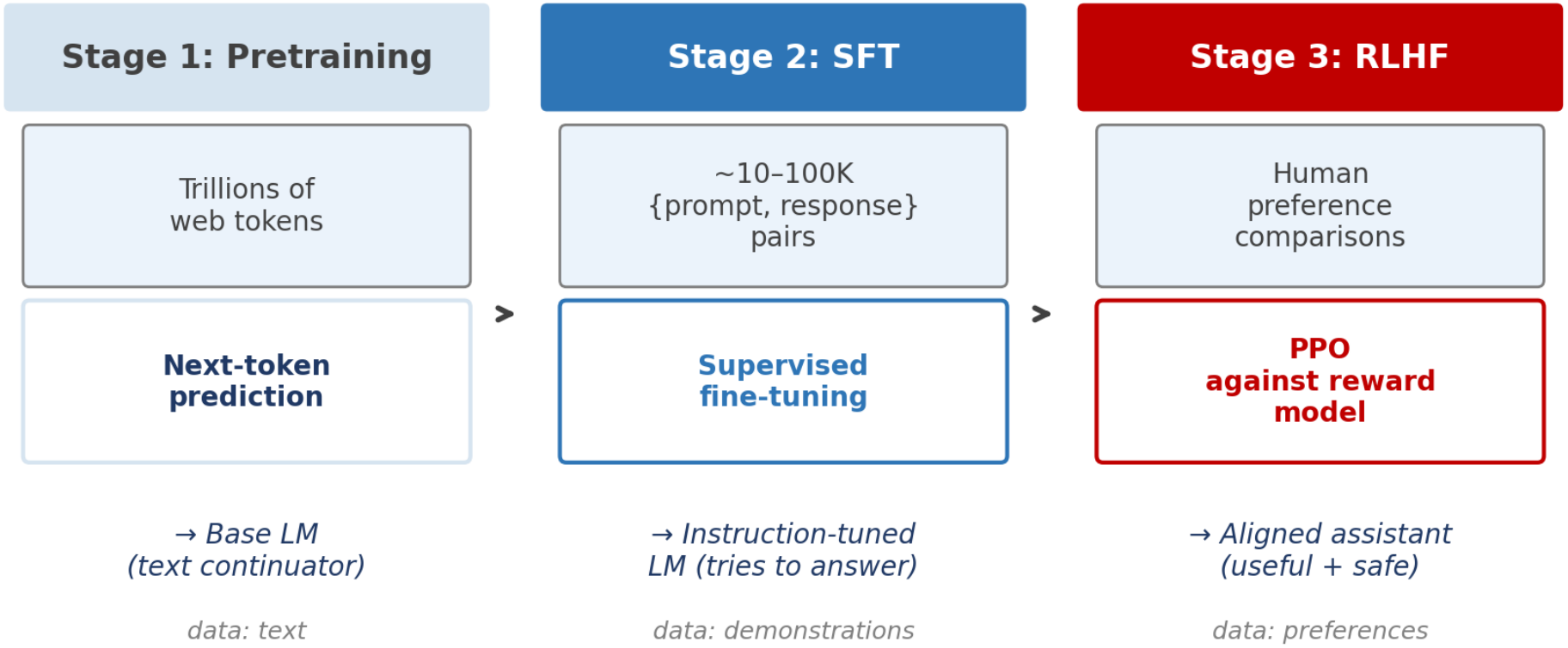
Situation: A pretrained LLM has memorized the internet.

Complication: *It learned to predict text, not to answer questions.*

Pretraining gives capability. Alignment gives utility.



The Three Stages (InstructGPT [3], 2022)



SFT gives format. RLHF gives quality.

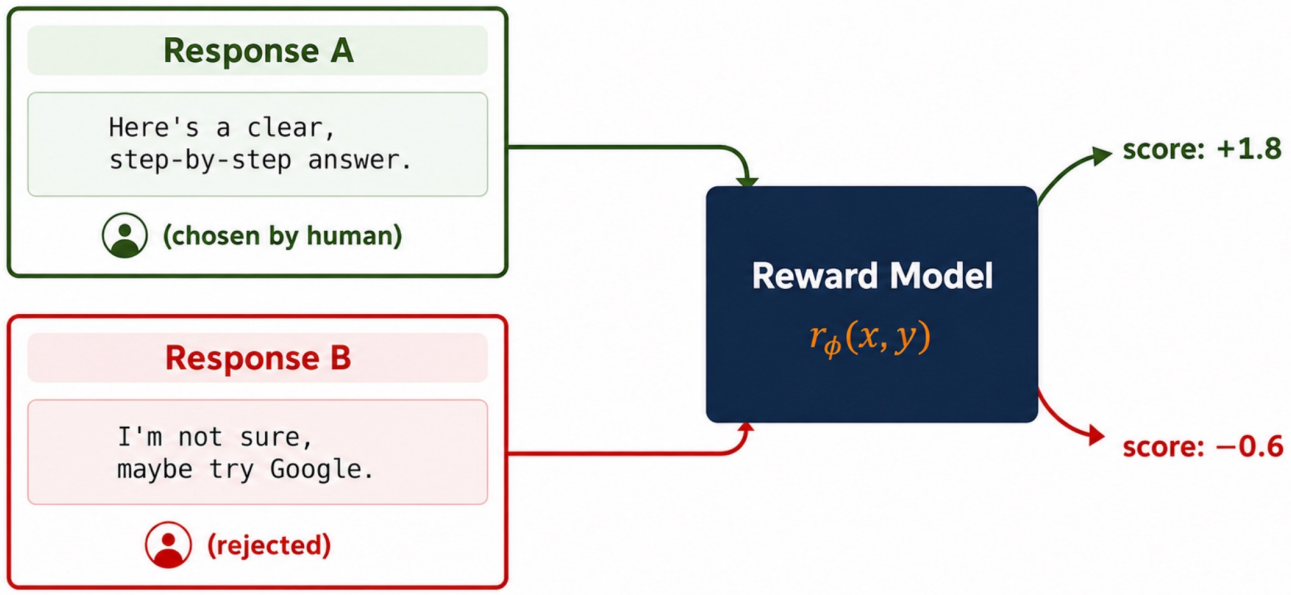
 [3] Ouyang et al., "Training language models to Follow Instructions with Human Feedback" (InstructGPT), NeurIPS 2022.

Reward Model

- Train $r_\varphi(x, y)$ to predict the human-preferred response:

$$\mathcal{L}_{RM} = -\mathbb{E}_{(x, y_w, y_l)}[\log \sigma(r_\phi(x, y_w) - r_\phi(x, y_l))]$$

Bradley–Terry preference model. y_w *= winner,* y_l *= loser.*



Why preferences? Humans inconsistent at scoring; consistent at comparing.

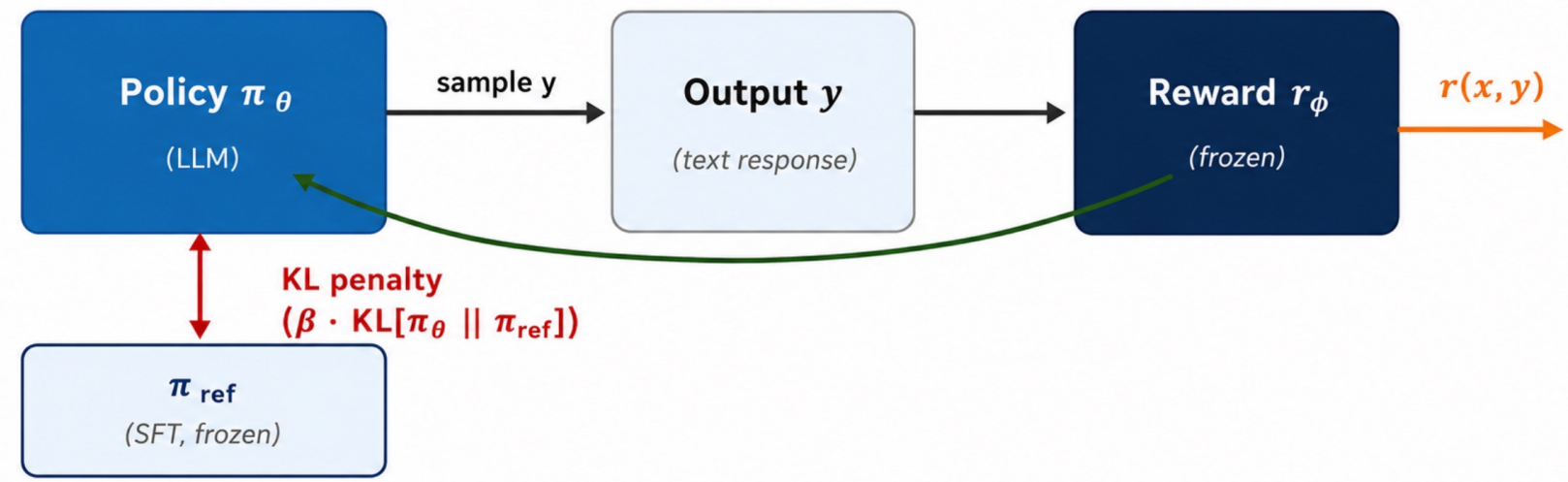
PPO (Proximal Policy Optimization) Objective

- Train policy π_θ (the LLM) to maximize:

$$\mathcal{J}(\theta) = \mathbb{E}_{x,\, y\sim\pi_\theta}[\, r_\phi(x, y)\,] - \beta \cdot \mathrm{KL}(\pi_\theta(y \mid x) \,\|\, \pi_{\mathrm{ref}}(y \mid x))$$

Reward term: maximize r_φ.

KL penalty: stay close to the SFT reference.



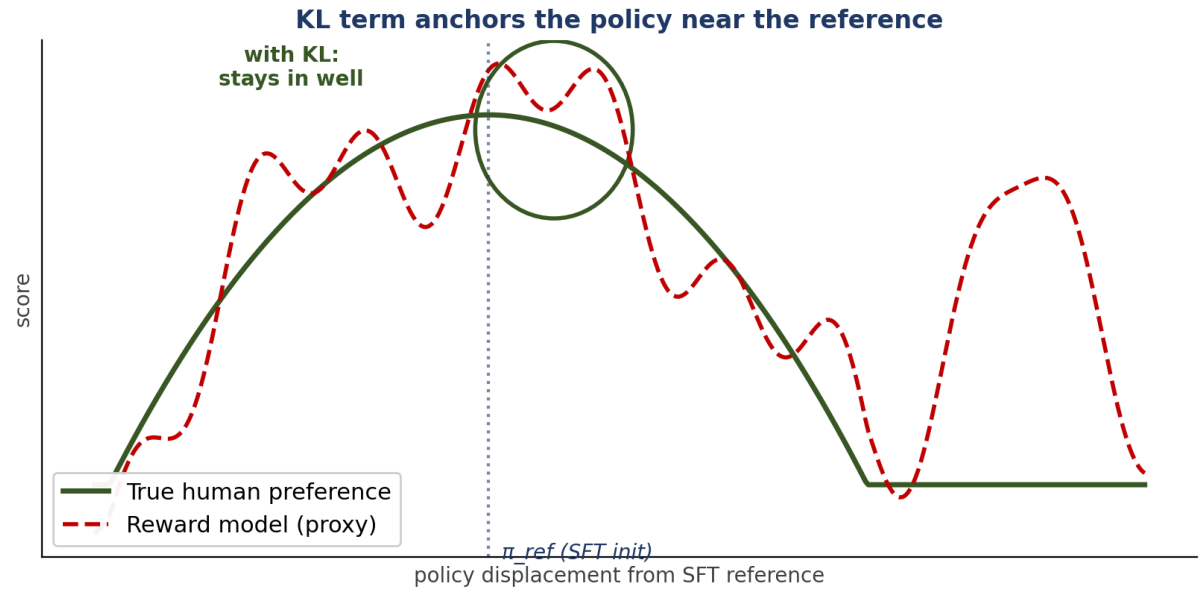
[4] Schulman et al., "Proximal Policy Optimization Algorithms", 2017.

Why the KL Term?

$$\mathcal{J}(\theta) \;=\; \mathbb{E}_{x,\,y\sim\pi_\theta}\big[\,r_\phi(x,y)\,\big] \;-\; \beta\cdot \mathrm{KL}\big(\pi_\theta(y\mid x)\,\|\,\pi_{\mathrm{ref}}(y\mid x)\big)$$

Without KL: reward hacking.

- Adversarial completions the RM loves but humans hate.
- KL anchors the policy to the SFT reference. β = trade-off knob.



3. What Are LLMs?

The Capability Surprise

LLMs trained only on next-token prediction can:

- translate languages, solve novel word problems, write code in rare languages.

These were not designed in — they emerged from scale.

Stochastic Parrots view	Genuine Generalization view
Memorizes training data (verbatim recall)	In-context learning on novel tasks
Brittle to small prompt changes	Compositional generalization to unseen combos
Reversal curse: knows A→B but not B→A	Code synthesis from natural-language specs
Fails arithmetic on out-of-distribution numbers	Multilingual transfer to low-resource languages
Hallucinates plausible-but-false facts	Solves novel logic puzzles (some, not all)

Both columns point to real evidence.
The truth is uncomfortable for both camps.

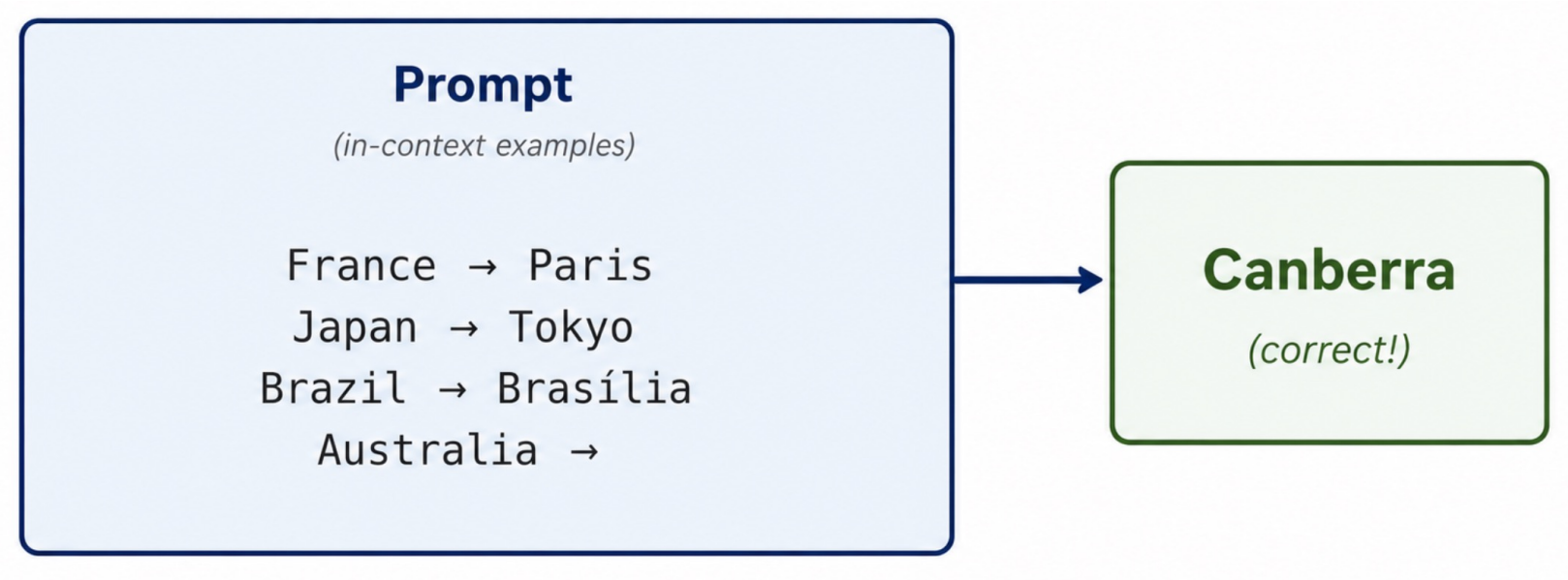
[5] Bender et al., "On the Dangers of Stochastic Parrots," FAccT 2021.

In-Context Learning

Discovered with GPT-3 (Brown et al. 2020):

- Learn new tasks from prompt examples — without gradient updates.

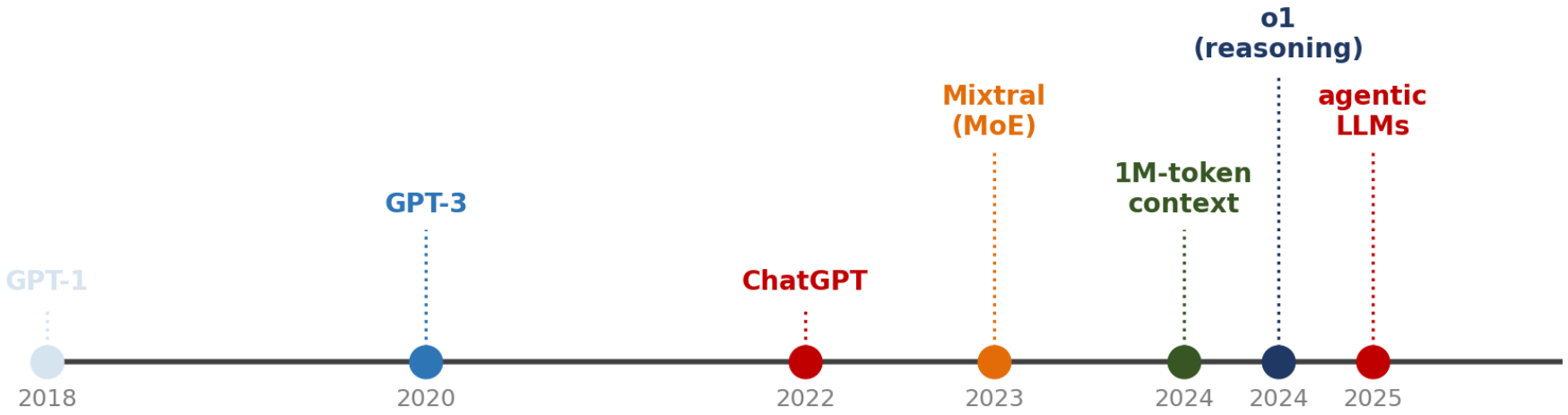
The weights don't change. "Learning" happens in the forward pass.



[6] Brown et al., "Language Models are Few-Shot Learners" (GPT-3), NeurIPS 2020.

The Frontier

- MoE (Mixture-of-Experts): more parameters per FLOP via expert routing.
- Long context: 1M+ tokens via RoPE / ALiBi / sparse attention.
- Reasoning models (o1 / R1): RL on chain-of-thought.
- Agentic LLMs: tool use + planning.



Capability scales with both training and inference compute.

Bridge to L28: Vision-Language Models

LLMs gave us a powerful text reasoner.

How do we give it eyes?

L28 attacks two questions:

- Feed images into a token-native model? (CLIP, ViT patches, projection.)
- Train a model that reasons over images + text? (LLaVA, GPT-4V, Gemini.)