How models are made

Key takeaways

1. Words become vectors, and a difference between vectors is a direction with meaning.
2. Pretraining predicts text; post-training uses reinforcement learning to shape behavior.
3. A benchmark is useful until its answers leak into the training data.

Common misunderstandings of how a language model is made

Which claim would you defend right now? Choose before the evidence.

"A good prompt teaches the model something it remembers for the next session."

"Training is one process: more parameters, more data, run for longer."

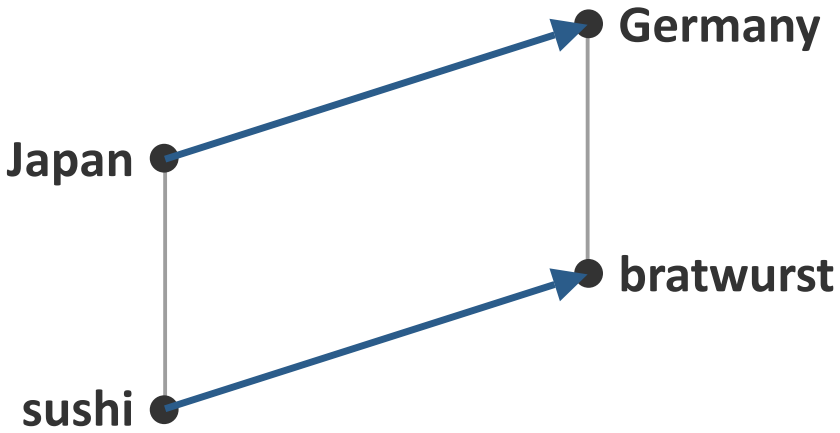
"A benchmark score describes the model, so it stays valid as long as the model does."

Keep your choice. Look for the mechanism or counterexample that would make it fail.

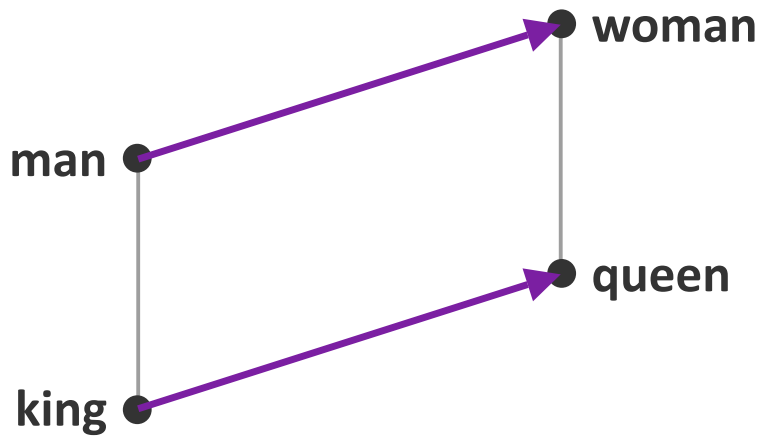
Tokens become vectors that carry meaning

GPT-3: 50,257 tokens × 12,288 dimensions, before layer 1 even runs

one plane through a 12,288-dimensional space (positions illustrative)



the same step: “German instead of Japanese”



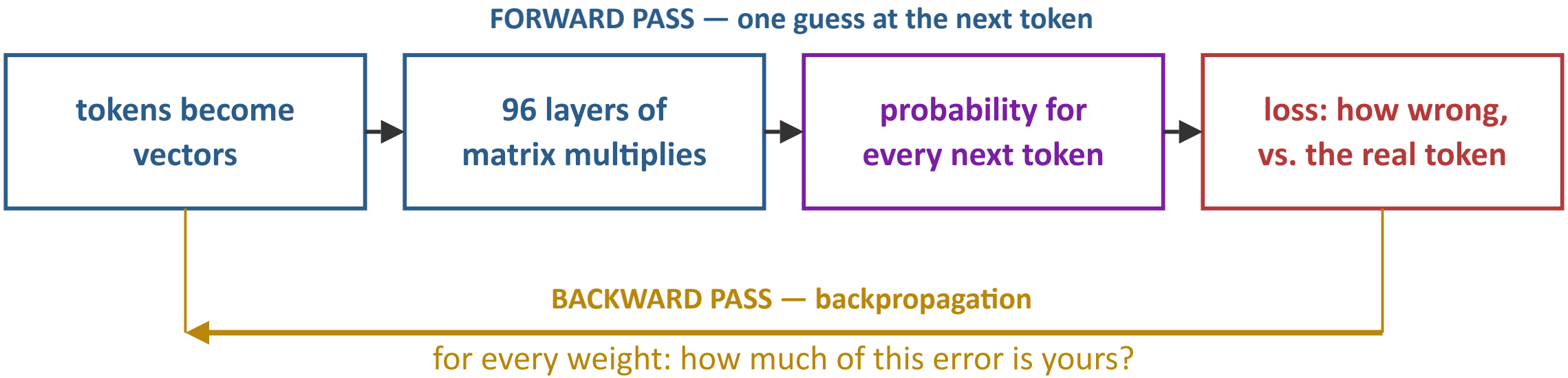
the same step: “female instead of male”

sushi − Japan + Germany ≈ bratwurst. Nobody wrote that rule. It fell out of predicting the next token.

source: Brown et al., arxiv.org/abs/2005.14165; 3Blue1Brown, “But what is a GPT?”, 3blue1brown.com/lessons/gpt

Training is blame assignment, repeated

Backpropagation computes each weight's share of one wrong guess



Then nudge every weight a little against its own blame, and repeat for trillions of tokens.

source: Rumelhart, Hinton & Williams, Nature 323 (1986), doi.org/10.1038/323533a0

Only two of the three stages change weights

Confusing them is why “just prompt it better” and “just train it” both over-promise

Stage	What changes	What it buys
Pretraining	weights learn to predict text continuations	language, code patterns, broad associations
Post-training	weights are tuned on demonstrations and preferences	instruction following and preferred behavior
Inference	the prompt changes activations for this call only	temporary task adaptation, no weight update

Your prompt, your context, and your tools all live in the third row. They change what the model does, never what it knows.

Post-training is reinforcement learning

The model is optimized against a reward, and the reward is a design decision

Step	What happens	What it optimizes
Supervised fine-tuning	humans write ideal answers for sample prompts	imitate a demonstrated style
Reward model	humans rank several model answers; a second model learns the ranking	a cheap stand-in for human preference
RL on the reward model	the model's own outputs are scored and reinforced	whatever the reward model rewards
Verifiable reward	reward = the tests pass, or the answer checks out	behavior a program can grade

The last row is why coding improved fastest: a test suite is a reward function that never gets tired — and it is the same signal the benchmark on the next slides uses to grade.

source: Ouyang et al., 2022, arxiv.org/abs/2203.02155

Preferred behavior is not the same as scale

Human raters preferred 1.3B InstructGPT over 175B GPT-3 on the study's prompt distribution

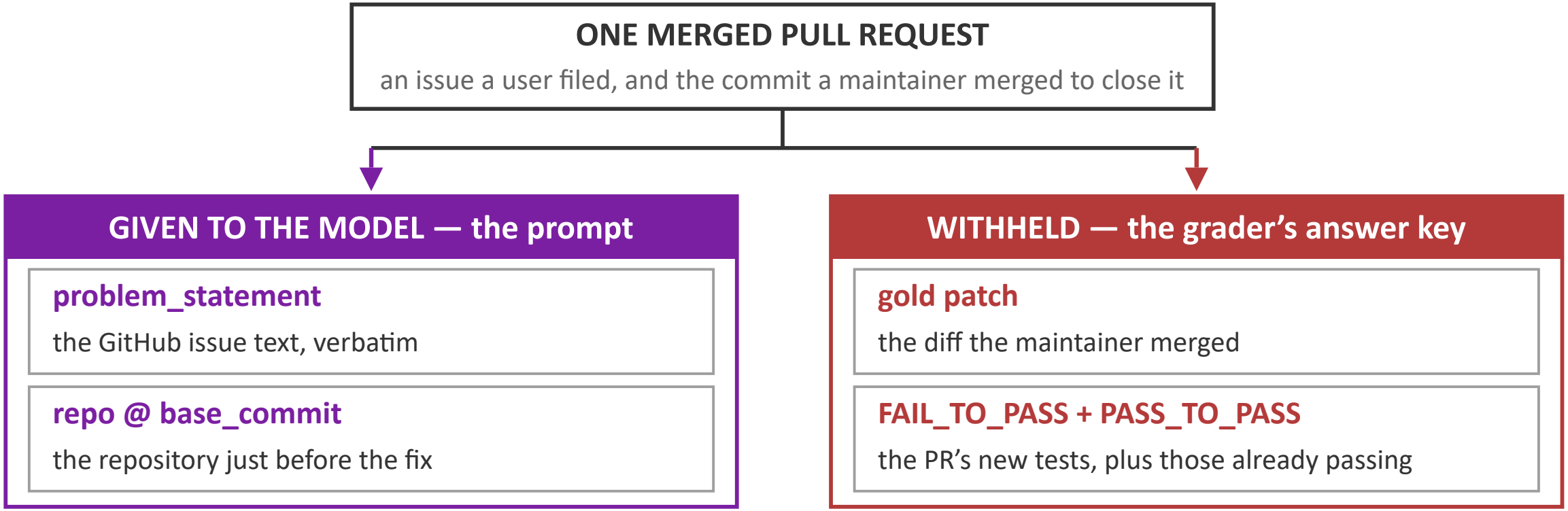
System	Parameters	Training emphasis	Observed preference
GPT-3	175B	next-token pretraining	less preferred
InstructGPT	1.3B	SFT + human-feedback RL	more preferred

- A model 130× smaller was preferred, because post-training changed what it was optimized for.
- This is about helpfulness on one prompt distribution, not capability in general.
- Parameter count and conversational polish are both poor predictors of task performance.

source: Ouyang et al., 2022, arxiv.org/abs/2203.02155

SWE-bench: a pull request, cut in two

"SWE" is software engineering: 2,294 tasks harvested from 12 popular Python repositories

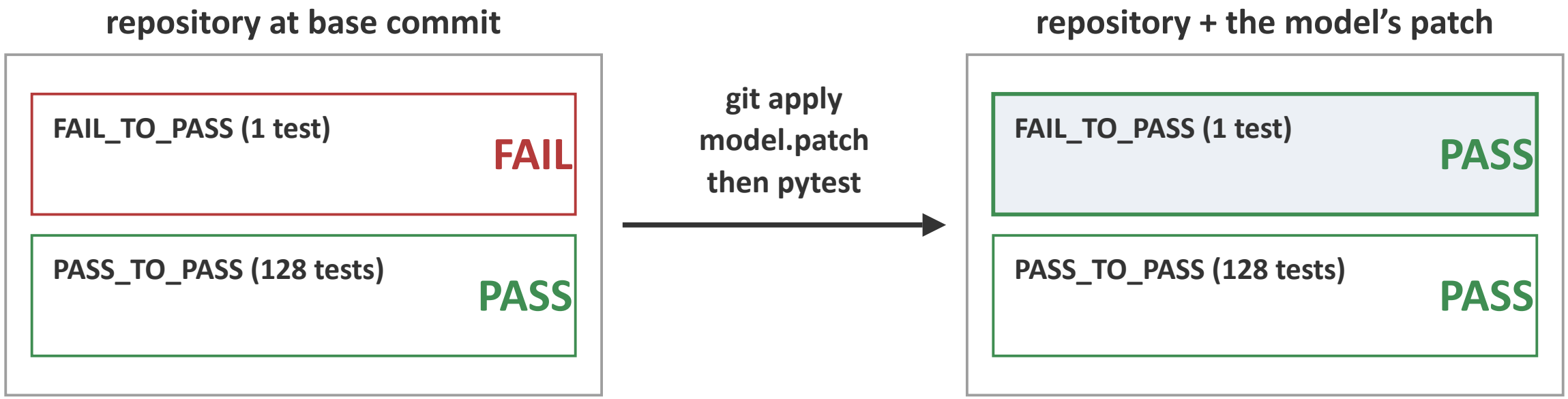


The model's whole output is a diff. The gold patch never grades it — it only told the harness which tests count.

source: Jimenez et al., arxiv.org/abs/2310.06770

“Resolved” is a two-sided test condition

Every FAIL_TO_PASS test must flip, and every PASS_TO_PASS test must still hold

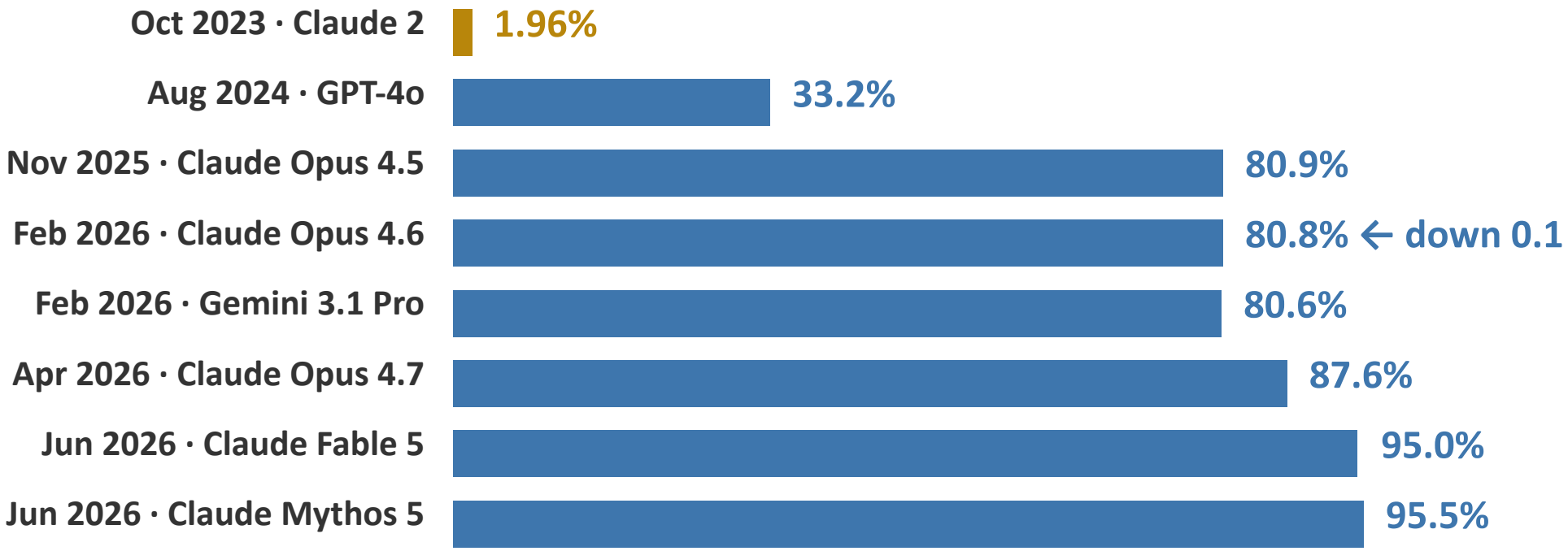


The shaded row is the one that must change. The plain row is the invariant: it is identical on both sides, and it is what a plausible but careless patch breaks.

One regression in PASS_TO_PASS scores zero. No partial credit, and no human reads the diff.

SWE-bench: 1.96% to 95.5% in three years

Each bar is a different harness, and the two colors are not the same test set



Amber is the original 2,294-issue test split; blue is the 500-issue Verified subset. Vendor-reported, different scaffolds — a trend line, not one experiment.

source: Jimenez et al., arxiv.org/abs/2310.06770; openai.com/index/introducing-swe-bench-verified/; leaderboard.steel.dev, retrieved 2026-08-12

The number stopped measuring SWE quality

February 2026: SWE-bench Verified's own coauthors stopped reporting it

What a deeper re-review found	Detail
Tests too narrow — 49 problems	reject a functionally correct fix over an implementation detail the issue never stated
Tests too wide — 26 problems	demand behavior the problem statement never mentions
Contamination	the issues come from popular repositories that are also training data, so no canary string helps
Verbatim recall	frontier models reproduced the gold patch or the problem statement from the task id alone

GPT-5.2 solved problems believed unsolvable — its chain of thought recalled an argument name from a later version of the repository that the issue never mentioned.

source: Latent Space, "The End of SWE-Bench Verified", 2026-02-23 — Glaese and Watkins, OpenAI Frontier Evals

Key Takeaways

1. A token becomes a vector, and the difference between two vectors is a reusable direction.
2. Pretraining predicts text; post-training uses RL to shape behavior. Neither is understanding.
3. SWE-bench measured real coding work for three years, until its answers leaked into training.