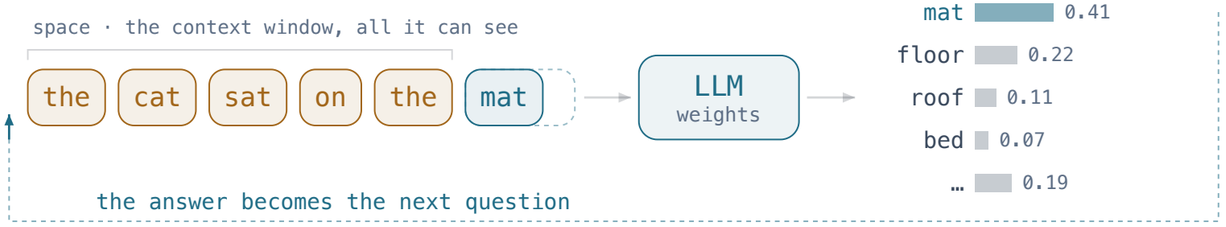
INTRODUCTION TO COMPUTER SCIENCE · WEEK 13 · STATION V

Machines

Now the question everyone actually came with. Today's AI is remarkable — and it is subject to every single thing we have established.

WHAT IT IS

A large language model is a machine trained on notation, asked for meaning.



tokens in · a distribution over the next one out · then a sample

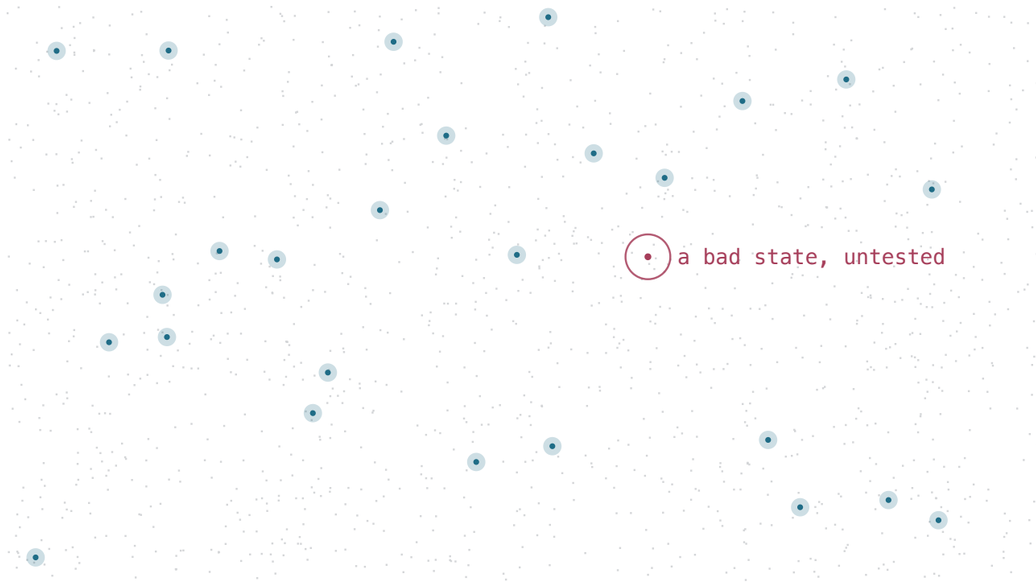
The name is exact. What it is trained on is *language*, so the signal is syntactic: which symbol follows which. Meaning is never handed to it. That this works as well as it does is the genuine surprise of the decade.

Context in, a probability for every next token out, one sampled, appended, and round again. The answer becomes the next question.

It cost megawatts for weeks; you run on twenty watts. Neither figure changes what kind of object it is: arithmetic on bits, a C* program in principle, which mipster would run.

STATION I APPLIES

Billions of parameters, each many bits. That space cannot be inspected, so the behaviour cannot be enumerated. Only sampled.



week 2, again

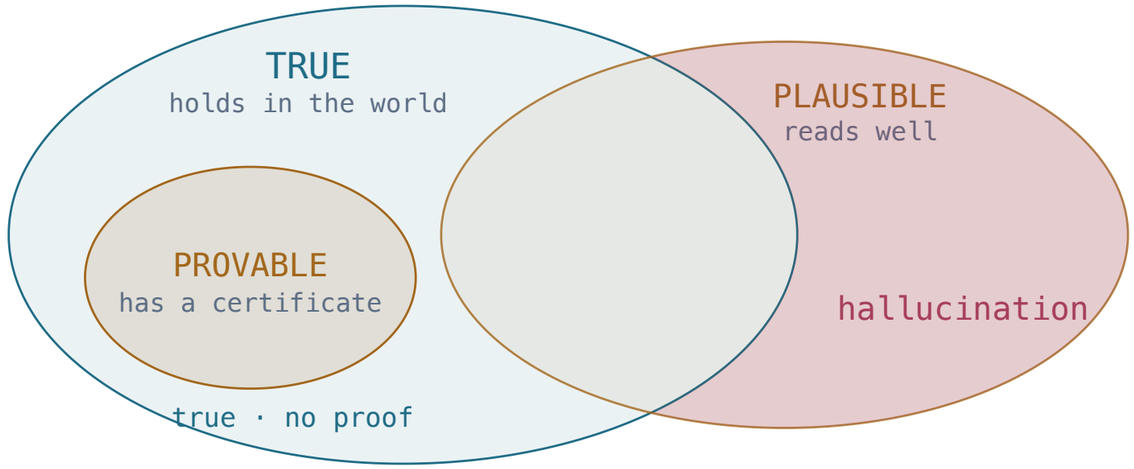
A model with a few hundred billion parameters is a point in a state space of the kind week 2 measured. Its behaviour — its output on every possible context — is one of week 7's infinite answer sheets.

So when a laboratory says a model was *tested extensively*, you know what that means. Whatever the tests exercised, the machine did. Whatever they did not, nobody knows.

Testing shows presence, not absence. Dijkstra, 1969, about programs. It was always about this.

THE OLD GAP, INDUSTRIALISED

A hallucination is a proof-shaped object that isn't true.



three tests · and they are not the same test

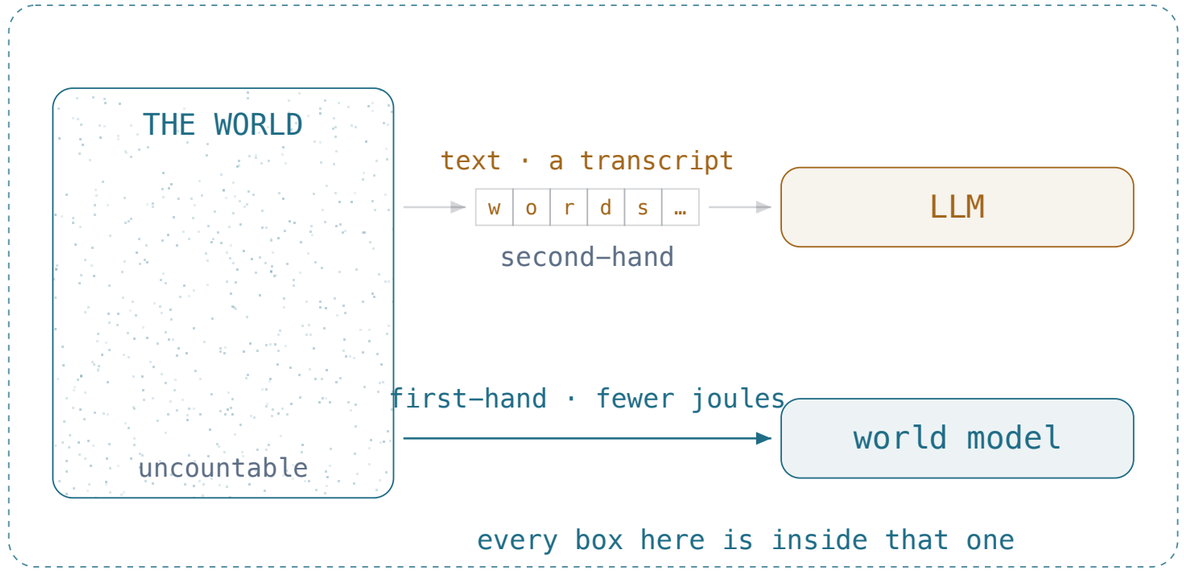
Fluency is syntax. Correctness is semantics. We built a machine of extraordinary fluency, so the gap between the two — week 7's gap — is something you now meet before breakfast.

Not a defect to be patched away: it is the proof/truth distinction at consumer scale. Harnad, 1990: symbols defined only by other symbols never touch the world.

So the durable response is not "trust it more" or "trust it less" but check it against something with a semantics — a compiler, a test, an experiment, a source, a colleague.

WHERE IT IS GOING

World models aim at meaning itself — and inherit every limit anyway.



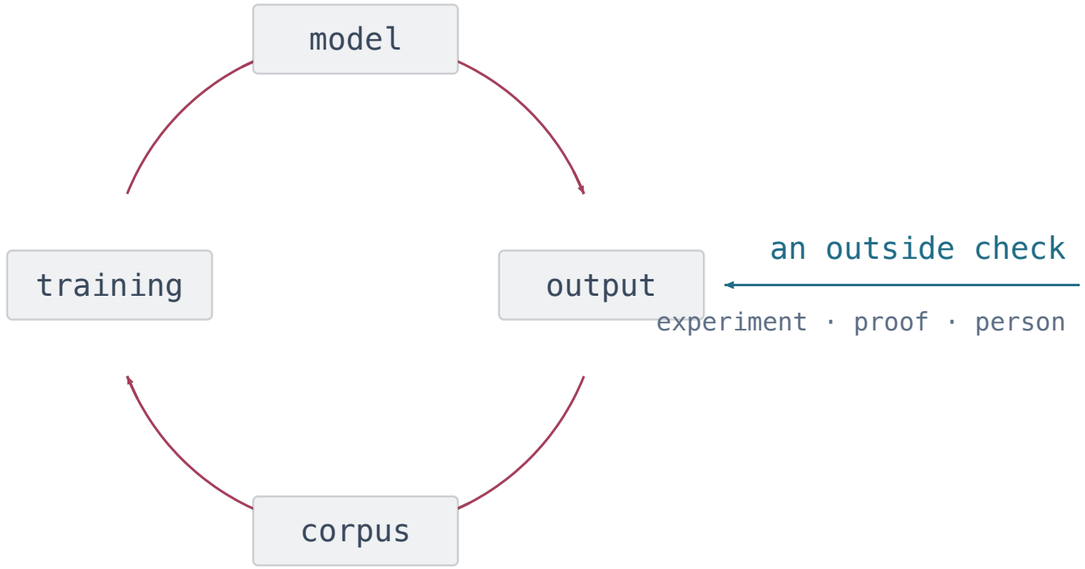
two routes to the same world

Not more text but a model of the world — state, dynamics, consequence. Predict what *happens*, not what is *said*: an attempt at real semantics, and a bid for efficiency. On a fixed budget efficiency is capability, so world models may well outperform language models outright.

And still a finite notation, sitting inside the world it models. Counting: models are countable, behaviours are not. Rice: "is this model right?" is semantic. Gödel: no self-certificate from inside. Cost: the state space did not shrink. Only the map did.

SELF-REFERENCE · AGAIN

The loops are already closed.



systems judging, training, writing systems

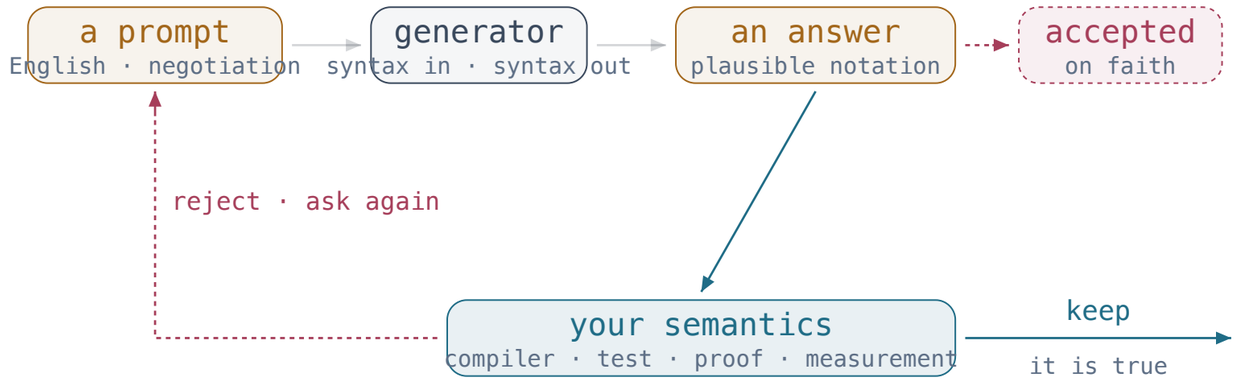
Models are trained on text that models wrote — with measurable degradation when the loop tightens (model collapse, 2024). Models grade models. Models write the code that trains models, and agents invoke themselves.

"Is this system safe?" is a semantic property of a program. Rice says: no general decision procedure. Not hard — impossible. And by Gödel's second theorem, no system this expressive can certify itself. An explanation of its own output is more output.

Which is not despair. It says exactly where to spend effort: on external, independent, bounded checks.

THE LOOP

A generator supplies notation. The semantics has to come from somewhere else.



what the machine supplies · and what it does not

Prompt in English, and what comes back is plausible notation. Unchecked, it is simply accepted — and then there is no semantics anywhere in the loop. Checked, it meets a compiler, a test, a proof, a measurement, and is kept or sent back round.

The fourth appearance of one theorem. Cantor's list needed an object built from outside it. Gödel's system needed a stronger one. Thompson's compiler needed a second compiler. The generator needs a check it cannot be.

```
$ ./selfie -c generated.c -m 1
./selfie: syntax error in generated.c in line 4: unexpe
```

WHAT ACTUALLY CHANGED

Generation got cheap. Verification did not.

BEFORE

Producing a draft, a proof sketch, a program, an image, a translation was expensive and therefore scarce. Scarcity did our filtering for us.

NOW

Production is nearly free and unbounded in volume. The filter has to be supplied deliberately — by specification, by measurement, by review.

Recall week 11: hard to find, easy to check. That is a description of a healthy relationship with a machine. Value migrates to the two ends the machine does not occupy: deciding what should be true (specification) and establishing that it is (verification). Both are acts of meaning. Neither is automated by better generation — and the better generation gets, the more they are worth.

BEFORE NEXT WEEK

Recommended exercises.

1. Read the Machines chapter.
2. Ask a chat bot for a C* program that reads a number and prints its factorial. Compile it. If it does not compile, note why; if it does, run it on three inputs.
3. Ask the same bot whether its program can divide by zero. Then run rotor and bitme on it with a bound of 500. Say which answer is a certificate.
4. Find a hallucination: an exact page number for a claim in a book you own. Place it in the three regions.
5. Write the four appearances of the one theorem behind the gate, and for each name what came from outside.

NEXT WEEK

So — what is intelligence? The definition, its two halves, why depth still matters, and what computer science is. Then the exam.