AI and Semiotics

Gene Callahan

Talk at the The Future of Technology Conference

Ludovika University of Public Service

Budapest, Hungary

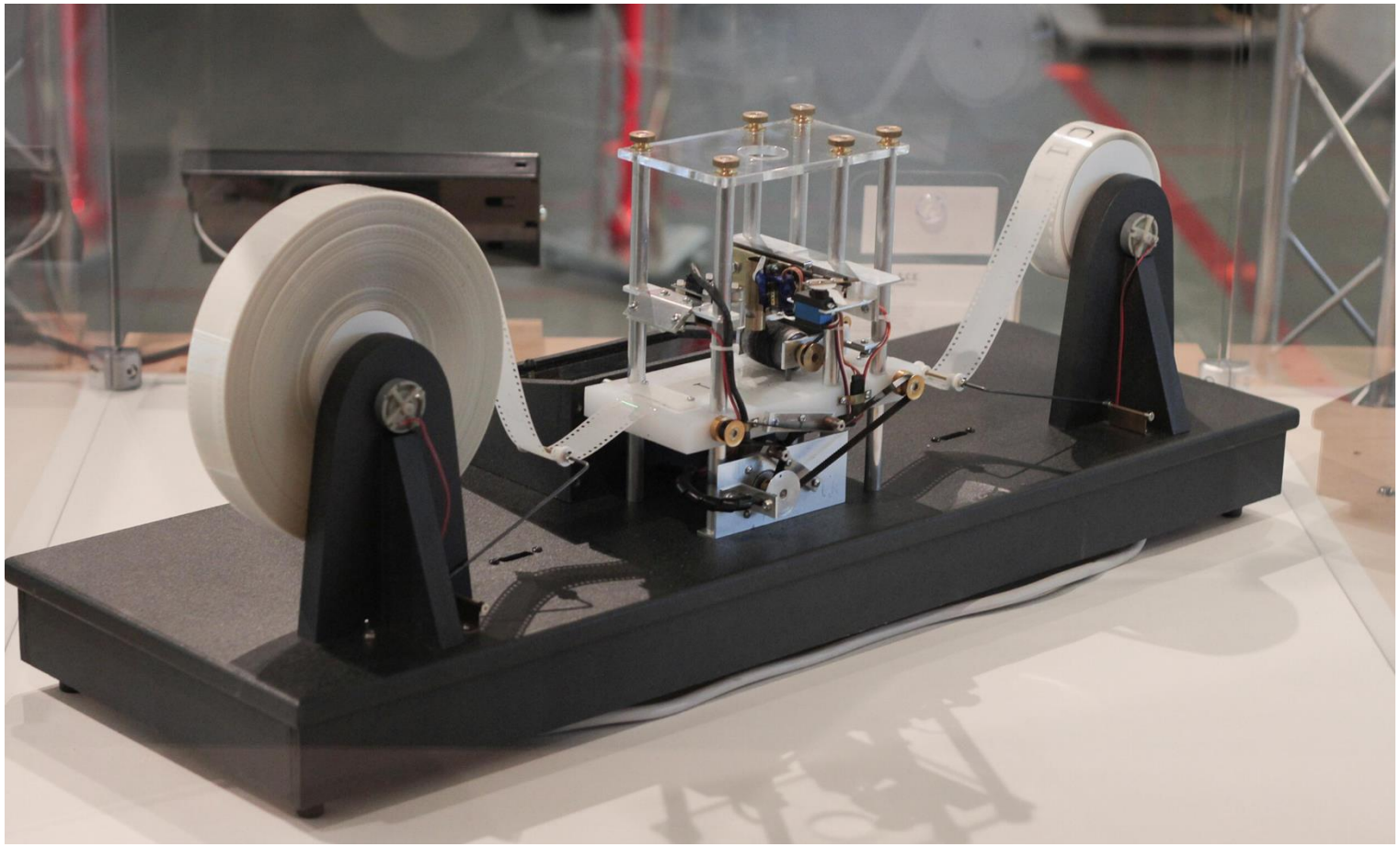
Just What Is “AI”

- Are computers becoming smarter than humans? Do we have to fear AI overlords directing humanity?
- Or is “AI” just a term for whatever the most impressive new software happens to be?
- Playing chess was thought to be cutting edge AI… until LLMs came along.
- Let’s look at what computers essentially *are*.

The Universal Turing Machine

- In the late 1930s, the British mathematician Alan Turing made an amazing discovery.
- He described a very simple device, consisting of a tape, a read-write head that could just write a fixed set of symbols onto the tape, and table of instructions as to what to do based upon what it read.
- The tape could be moved back and forth based on the instructions, and could replace a former symbol with a new one, or put a symbol where none had been.
- And that’s all!

The Universal Turing Machine



The Universal Turing Machine

- Turing's amazing discovery is that an instance of this very simple machine, called a *Universal Turing Machine*, could compute *anything* that any mechanical device can compute.
- Turing's PhD advisor, Alonzo Church, achieved a similar result with his *lambda calculus*.
- It was shown that their models were formally equivalent.
- This is perhaps the most important single finding in CS: the *Church-Turing thesis*.

The Church-Turing Thesis

- What their discoveries mean is that no advance in hardware or software can ever expand the universe of computations that any UTM can *theoretically* achieve: an Apple I from 1976 is formally capable of performing any calculation that a giant LLM server farm can today.
- Of course, the Apple I might take millions of years to compute something the server farm computes in a few seconds. So as a practical matter, there is an enormous gulf here.
- But remembering this principle will help us to understand what comes next.

How Does a Digital Computer “Compute”?

- The insight that led to modern digital computers was that any physical system that can be engineered to be in one of two states (the “ones” and “zeroes” we hear about) can be used to create a UTM.
- It is quite possible to build a water computer or a pneumatic computer.
- Why do we build mostly electronic computers? Think about how long it takes from flipping a switch to see your light turn on, to how long it takes from turning on the hot tap to get hot water from your sink.
- But we know from the Church-Turing thesis that a water computer could compute anything an electronic computer could.

Building a Water Computer

- We will assume that we have the plumbing technology to construct the system at hand.
- We are going to build a system of pipes, that can contain either hot water or cold water, that those two states are clearly distinguishable, and that they are the only thing I machine will care about in terms of the water flowing through it.
- At certain points, two of these pipes will come together in an engineered device called a gate, which will output either hot or cold water based upon what arrives at the gate. Let’s look at what we mean by a gate.

Building a Water Computer: Gates

- Let’s build a logical AND gate for our water computer!
- We’re going to interpret hot water coming out of the gate as meaning TRUE.
- And we’re going to say that we only want our gate to output TRUE if both pipes leading into the gate were feeding in TRUE, in other words, hot water.
- [MATT: here a series of slides showing cold-cold, hot-cold, cold-hot, and hot-hot going into a triangle, and [cold-cold] -> cold, [hot-cold] -> cold, [cold-hot] -> cold, and [hot-hot] -> hot coming out.

Building a Water Computer: Gates

- But there is nothing intrinsic in the physical set up that determines it to be an AND gate: that is *our* interpretation.
- With the exact same physical set up, but where we choose to interpret cold water as TRUE, the device becomes the opposite gate: a NAND gate.
- So, even with the simplest of logical operations that occurs inside a computer, what the result *means* depends upon how *we* want to interpret it.

Building a Water Computer: Gates

- But there is nothing intrinsic in the physical set up that determines it to be an AND gate: that is *our* interpretation.

Building a Water Computer: Gates

- We don’t even have to interpret the physical set up as a logic gate.
- We might instead consider it… our water park’s coin flipper.
- Hot is heads and cold is tails!
- If we designed the right “program” at the top of the hill, we can simulate random coin flips that produce half heads and half tails.
- The “program” could be a set of instructions to people with pails of hot and cold water at the top of the hill.

Building a Water Computer: A Network of Pipes and Gates

- If we buy a bigger hill, we can set up a working, water-based computer.
- It will be *very* slow!
- Let’s say we have as few as 32 open pipes at the top of the hill where we can provide our computer with input. That gives us 4 billion possible combinations of “code” and “data.”

Running Our Water Computer

- Thirty-two people at the top of the hill input a program and some data it should process by pouring a combination of hot and cold water into the open pipes at the top of the hill.
- We are at the bottom of the hill, and see that a pattern of TRUE and FALSE, or ones and zeros, or hot and cold water, is our computer's output.
- What is the meaning of this output in front of us?

Running Our Water Computer

- Unless we know the *meaning* (if any) that the people at the top of the hill assigned to their inputs, we cannot have any idea at all what the meaning of the output of our computer is.
- We just see a bunch of hot and cold water coming out of pipes.
- The inputs at the top could've been intended by the people with buckets to calculate a batting average, or generate a snippet of music, or pick out a good chess move.
- Whether the program is "correct" is also entirely based upon what the inputs *meant* to the people doing the inputting.

What Does This Mean?

- No computer program in and of itself has any intrinsic meaning.
- All meaning in the running of a computer program are input by humans at the start of the program, and interpreted by humans, in light of the meaning of the inputs, as the meaning of the output.
- In Peircean terms, a computing machine has no *thirdness*.

But Don’t LLMs Change All This?

- Surely, something as sophisticated as a modern LLM is an entirely different entity than our silly water computer, right?
- Similar claims have been made for expert systems, and machine learning, and object-oriented programming, and distributed computing.
- All such claims ignore… *The Church-Turing Thesis*!
- Our water computer can calculate *anything* that any mechanical calculating device can calculate.

LLMs Are a Great Technical Achievement

- But acknowledging that achievement has no metaphysical implications!
- What is really going on in machine learning: it is a search of program space for a program that is “good enough” at some task, e.g., answering users’ questions off the Internet.

C.S. Peirce

- Chemist, surveyor, mathematician, logician, astronomer and founder of modern semiotics and pragmatism.
- Bertrand Russell: "certainly the greatest American thinker ever.”
- “As early as 1886, he saw that logical operations could be carried out by electrical switching circuits.”

Secondness

- Peirce developed the concept of *secondness*.
- When a rock tumbling down a hill meets another falling rock and deflects its course, we have a purely *dyadic* relationship: neither rock *means* anything further when hitting its companion.

Thirdness

- Peirce pointed out that human language, on the other hand, is intrinsically a *triadic* relationship.
- If I shout to you “A mind-reductionist is creeping up behind you!” I am not merely sending some sound waves to your eardrums: I am also *referring* to a *third* thing: the reductionist.
- [MATT: a triangle with the corners being “sign,” “object,” and “interpretant.” A man (interpretant) interpreting the word “dog” (sign) to mean the actual furry animal (object).]

Thirdness and AI

- Think about our water computer: the water flowing down the hill, means nothing by itself. It is only a shared meaning between the inputters at the top of the hill and the readers at the bottom of the hill that gives the program being run what meaning it possesses.
- The bits that represent your department's accounts for the year could just as well be bits that represent a musical composition or an image to display.