# THE CURTAIN

## LUKE SAWCZAK

First published in Queen's Quarterly, Spring 2022

Software, says Jaron Lanier, is infinitely fun when it's a small project and you have complete control, but infinitely draining when it's at corporate scale and your job consists of linking large, boring subsystems.

This is why I don't work in software.

### I

I made a cake cutter applet (tools.unfamiliarplace.com/cake), because I can never visualize the cuts required to make *n* slices of a cake where *n* is the number of people and *n* is greater than 4 and is not a power of 2 or equal to 6. In my applet, you get a little interface where you choose the value of *n* and it draws the number of slices. I got stuck at the drawing slices part, but then I remembered a math teacher of mine who said that trigonometry is used in computer graphics, so I used sine and cosine to work out the hypotenuse of a triangle from the centre to the edge of the cake. I was very proud of this, though it turns out the principal joy of the app is the fact that if you hold the up arrow on the number spinner, the new lines are drawn so quickly that it appears the cake grows an ever-denser black hole from the middle outwards, while generating moiré patterns in the empty space. I made the limit on the number of slices ridiculously high so as to maximize this effect, and I let you choose the colour of the cake and the cuts so as to allow fanatics to make nice designs.

I always want to excitedly show this app to people. It's of very little use, of course, even though there are (toggleable) arrows and labels so you can know the order and direction of the cuts to properly divide your cake. Everyone quickly discovers the black hole thing and gets a laugh out of the very idea. This is why I write software.

### II

There are two paths into programming, I realized as I watched the students in my labs at the University of Toronto handle things like the trigonometric cake cuts with ease. You can enter it either through math or language. My way in is language.

I studied linguistics, and I continue to compulsively learn languages, seeking out any opportunity to hear non-English, at the risk of boring friends and family. Programming is absolutely about eloquence and fluency, and conversely, language is absolutely about structure and design. (I have a bias to suspect that those who come to programming through math alone write inelegant code. This isn't fair, of course. There are many math-oriented programmers who also appreciate style, Don Knuth first and foremost.)

I do love math; I just don't understand it. When I was in high school I carefully worked out by trial and error that given the square of an integer x, you can find the square of $x + 1$ by calculating $x^2 + 2x + 1$. Take $3^2 = 9$, add $2(3) + 1$, and you get $16=4^2$! I shared this discovery with my math teacher, who proved it in ten seconds by expanding $(x + 1)^2$, to my dismay.

## III

My younger brother started tutoring math some time ago. He loves inventing problems and scenarios to keep the kids entertained, and they keep coming back to him. Early on, he had the idea (convergently evolved elsewhere) for the word-searching game Boggle™, but with math. So I made Math Boggle for him (tools.unfamiliarplace.com/mathboggle). The board shuffles numbers and operations, and you "spell" equations that, for the purposes of tutoring, you have to calculate. You can change the range of the numbers, the proportions of each operation (why not focus on division today?), the bias towards high factorability versus prime numbers, and so on. The board shuffles with a shaking animation and sound, and you can rotate it with the sound of spinning a chessboard on a wooden table, in case looking at it from another angle helps you find more equations.

At first my brother wondered whether the rotating was really necessary, but soon the students confirmed that hearing the "whuff" and watching the numbers counter-spin to the board so they remain upright is half the fun.

## IV

I like good design. I love object-oriented programming, even if it has its detractors, and even if their objections are valid. But, damn it, it's so fun to reinvent simple Point, Line, Circle, Cake, and Knife classes when you want to make a little applet. Lanier would agree: coming up with the basics of a working Circle class is a barrel of laughs, even though the canonical implementation is probably made by Adobe, has 90,000 lines of code, is the seventh-level descendant in a hierarchy of interfaces, and it refers to 19 other classes to coordinate its appearance on the screen. That one I wouldn't touch with a ten-foot pole.

On the other hand, I use Photoshop too. And I'm glad that behind the circle-drawing tool is an arcane Circle or Ellipse or Ellipsoid. The appearance of simplicity is compatible with, or even depends on, underlying complexity. That was what physicist Richard Feynman meant when he said he can't explain magnetic forces in terms of rubber bands: the real substructure is nothing like the surface, and an analogy with rubber bands would give the wrong impression. We want a world made of simple atoms and axioms, but the world contains simple things like rubber bands and monasteries only because of the complexity under the hood – the total non-rubber-band-ness of it.

## V

I've just finished *The Brothers Karamazov* – that's why I mentioned monasteries, not because I thought they were a perfectly ordinary thing to bring into a sentence alongside rubber bands. But in TBK I was introduced to the scientist Claude Bernard and to his horrifying vivisection, but also to his work advancing the scientific method in biology.

In Bernard's time it was common to apply statistics to biology. One could observe that in this or that illness, such and such a percentage of people died. If a certain curative were applied, one could observe a certain rate of convalescence. To Bernard, however, this did not amount to scientific knowledge nor contribute to meaningful practice. Mathematics may have a role in every field, he said, but applying averages before understanding the mechanisms yields "only apparent accuracy.… Qualitative analysis must precede quantitative analysis."

## VI

In the software world, math is closer to the centre, and to the money, than ever before, but the scientific method is not. We've excitedly given up on trying to understand many problems, opting in favour of a black box that "just works."

A typical example of the path comes from machine translation. A natural approach might be to take the words' meanings in a source language dictionary and find their equivalents in a target language dictionary; you could create a mapping of source to target words in this way, i.e. a bilingual dictionary. If you're clever, you could avoid limiting yourself to such pairs and create an intermediate dictionary of abstract meanings, mapping each language bidirectionally to the abstraction so you can discover that "smile" = 😊 = "sourire," the 😊 serving as a pivot or hinge between any pair of languages. Next, you need the grammar. In English we say "I love you," but in French "Je t'aime," so we need to (a) identify the subject, predicate, and object and (b) swap the order of the predicate and direct object in translation, according to a

similar pivot-mapping of abstracted sentence roles. This is a simplified version of what's called rule-based translation.

This approach turns out to be somewhat naive. *Does* every word have an equivalent? That's problem one. And *do* we understand the underlying structure of language? Do all languages even share commensurable grammars? That's problem two. This is an active line of inquiry – in fact it encompasses most of North American linguistic research as kicked off by Noam Chomsky over 60 years ago – and while much progress has been made, the end is not in sight. Systematically modelling languages turns out to be massively complex.

That's a research crop ripe for the picking, but as a method of practical machine translation, it turns out to be woefully inadequate. Rule-based translations usually feel somewhat stretched, and are notoriously insensitive to context. This barrier was recognized early on, and so statistical methods began to be invented, and then neural methods, which are also fundamentally statistical.

The neural method is easy to understand, although its implementation can be sophisticated. The basic principle is to train a system by showing it carefully labelled data, such that through trial and error, it discovers correlations between individual properties of the data and the labels.

A correlation's weight depends on its ability to yield accurate predictions about previously unseen data. When the training has identified the strongest correlations, such that the predictions are acceptably accurate, the model can be released for use on data we haven't yet seen. In machine translation, this means that words and phrases that have been translated a certain way in one context can probably be translated that way in another if the contexts are similar enough.

The results of this type of method can be seen in the rapid pace of growth of AI in all fields. It yields translations that are usually much better than rule-based ones. Not always, though – sometimes it breaks down.

When it breaks down, it does so in strange ways. In the famous Jeopardy! games, IBM's Watson computer displayed its top three guesses for the answer. What was curious to me was the steep drop in plausibility between the top one or two guesses and the third. For "This mystery author and her archaeologist hubby dug in hopes of finding the lost Syrian city of Urkesh," Watson guessed: Agatha Christie, Sue Grafton, and … Syria. Again, I was recently uploading photos to a service that uses machine learning to optimize portraits, and while most were subtly improved, some blue backgrounds turned green at random, and some skin colours changed hue dramatically. On thispersondoesnotexist.com, most AI-generated faces are highly believable, but when they're not, it's because people have second heads or glasses made of eyes.

The retort goes: one or two more generations, and the kinks will be straightened out. But it's not the accuracy that's concerning – it will only ever be apparent accuracy. The key is that these errors are not just wrong, but implausible; not only the results, but the steps, are nonsensical. It's not the wrongness, but the lack of method that concerns me.

When we resolve these problems, the resolution will be no more logical than the problem. The devil's bargain that we make with machine learning is Bernard's statistical success without any actual advance in science, in material knowledge. Quantitative analysis will supersede qualitative analysis.

## VII

"So what? Suppose we apply it to Bernard's field of medicine; if we give up understanding the mechanism but we cure more people, isn't that a good trade?"

Bernard wrote that a theory is validated only by the confluence of much evidence, but Karl Popper a century later went a step further when he realized that science actually rests on falsifiability. It's not evidence for a theory, but the possibility of evidence against it, that distinguishes science from pseudoscience.

On the vacuousness of merely accumulated evidence, Popper cites an anecdote that would have appealed to Bernard: A man suffering from mental illness is brought to a famous psychiatrist. The psychiatrist asks him a few questions, then rapidly diagnoses the case as schizophrenia.

"But how can you possibly have determined that so quickly?" Popper asks the doctor.

"Simple," he replies. "I've seen it a thousand times."

"Then I suppose this makes a thousand and one."

The same might be said of machine learning. It is self-confirming. The original source of its accuracy is the labelled data that comes from humans. On data identical to the labelled data, we get overfitting – when there's nothing new produced by the machine, only what we told it. Conversely, the less the data resembles the labelled data, the more *under*fitting there will be. New cases must be somewhat like the old or they will fail. And the issue is that when we get to that new disease, and the AI-selected cure doesn't work, we have no insight into the cause or a path forward.

## VIII

AI is awfully good at imitating reality, and getting more convincing all the time. This would not be dangerous if we ourselves were machines following rigid rules and doing strict comparisons. But in fact we're susceptible to change; we learn from what we see.

Lanier argues in *You Are Not a Gadget* that we are naturally forgiving of software models. Many of my generation have had the experience of looking back at a game we played in the 1990s or early 2000s and being surprised to discover that the graphics don't look realistic at all. As kids, we believed in them implicitly and were even awes-

truck by their resemblance to reality. Only by comparison with something closer to the ground truth do we notice the shortfalls. This subtle acceptance is a threat if and when software claims not simply to capture some aspect of the world with which we can scientifically compare it – say, the way light reflects off water – but to capture some aspect of ourselves which is not so easy to prove inaccurate. *Is* this generated dialogue really how people talk? How we act? How we think?

Alan Turing proposed an experiment wherein computer-generated conversation is compared with that of a human and is benchmarked for how many humans fail to distinguish the two. Today, we propose AI-generated faces, voices, conversations, movements. The danger is that for us to consider that the machine has passed the test, it's not necessary for it to be human-like; it's sufficient for *us* to become machine-like and fail to notice the shortfalls. Each new example, like Popper's psychiatrist, confirms the impression and impairs our ability to distinguish the real thing from the statistical average.

I'm reminded again of Dostoevsky's theme of psychological plausibility. What is within the range of normal human behaviour, and what's pathological?

## IX

One of my students gave a wonderful presentation in which, to my surprise, he described AI not as a branch of computer science, but as its opposite. This makes me think of my still-unfinished prose-to-poetry converter called Odeon. You put in some prose, turn a few dials to influence rhyme, rhythm, and freedom to substitute synonyms, and it tries to output poetry. For example, here's a sentence from *The Hobbit*: "Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat …"

One configuration of the converter that privileges rhythm produces:

```
Not a nasty, dirty, wet hole,
Filled with the ends of worms
And an oozy smell, nor yet a dry, bare,
Sandy hole with nothing in it to sit
Down on or to eat.
```

Another configuration that privileges half-rhymes produces:

```
Not a nasty, dirty, wet
Hole, filled with the ends
Of worms and an oozy smell, nor yet
A dry, bare, sandy hole with nothing in it to sit
Down on or to eat.
```

For half-rhymes, I wanted to score the similarity of words so that the user can choose precisely the threshold they're willing to accept. To do this, I had to write a little engine that relies on a breakdown of syllables and phonemes and segments with place of articulation, voicing, and other details drawn from my linguistics background. It was much more work than the feature really warrants. And you could probably get much better results with machine learning; it would be a trivial, not to say cheap, process to strategically insert line breaks in a bunch of prose and code the contexts for a handful of properties by hand or using Natural Language Processing to yield a labelled dataset, then train a system that randomly inserts line breaks and scores the results.

But I wouldn't change my approach for anything. For this amateur, there was joy in coming up with my system and a sense of satisfaction. I know how it works and why it works, and when it fails, I can determine what the problem is; I have directions I can follow to address it or to innovate. The input is given a structure that interacts with the rules to produce a predictable output, and the method can be improved on by the application of reason and theoretical and empirical knowledge.

## X

It might be argued that machine learning is a scientific instrument, just a very subtle one. Though it proceeds by trial and error, it hits on connections that are really there, and if I'm shown an ad based on a million data points of Internet usage, it's my limitation if I perceive it as random. Machine learning is on the same side in the fight against entropy: it takes chaos and produces order.

But machine learning is founded on correlation, not on causation. It produces only the illusion of order, or rather, any order in it is what we bring, while in reality it remains a chaotic rearrangement of the materials. I don't believe in machine learning's ability to discover true mechanisms.

The results of processes do not map uniquely to the processes they result from. A neural network can discover patterns in pixels – hundreds, thousands, or millions – and generate new arrangements of those patterns that look like faces to us. But it will have learned nothing of the DNA that actually determined the shape of the faces it was trained on. Crucially, we can carefully decode proteins in helices to understand what each strand contributes, but it's impossible to look inside a massive AI like GPT-3 and know why it does what it does.

Scientifically speaking, this is a step backwards. In one of his lectures, Feynman compares a modern theory of physics with a medieval one as they apply to motion in the solar system. He traces the consequences of the theories, and finds that they both account for the phenomena; in that sense they're equally accurate. Up to one point, that is. Where the medieval theory doesn't account for one particular motion of a planet, it posits an angel beating its wings in space to provide some of the propulsion.

Two theories can correspond to what we can see on the surface, while only one of them provides fertile ground for scientific progress.

## XI

I make simple tools, designed for humans to enjoy. I bring my own ideas to them, and I fail as often as I succeed. But the results are hard-won and unique. It's difficult and slow to make things with a thorough understanding of what they are, but it's vastly more worthwhile.

I prefer Illustrator to Photoshop, vector to raster, shapes to pixels. You can imagine a Photoshop neural filter to, say, convert sharp, square corners into smooth, round corners. But give me the squares, and let me modify the border radius. Give me a collection of abstractions with structure and method, rather than drawing the curtain and then raising it up to show that a magic trick has been performed – even if the magic is quicker at producing impressive effects than the science.