Franken-algorithms: the deadly consequences of unpredictable code

The death of a woman hit by a self-driving car highlights an unfolding technological crisis, as code piled on code creates 'a universe no one fully understands'

The 18th of March 2018, was the day tech insiders had been dreading. That night, a new moon added almost no light to a poorly lit four-lane road in Tempe, Arizona, as a specially adapted Uber Volvo XC90 detected an object ahead. Part of the modern gold rush to develop self-driving vehicles, the SUV had been driving autonomously, with no input from its human backup driver, for 19 minutes. An array of radar and light-emitting lidar sensors allowed onboard algorithms to calculate that, given their host vehicle's steady speed of 43mph, the object was six seconds away – assuming it remained stationary. But objects in roads seldom remain stationary, so more algorithms crawled a database of recognizable mechanical and biological entities, searching for a fit from which this one's likely behavior could be inferred.

At first the computer drew a blank; seconds later, it decided it was dealing with another car, expecting it to drive away and require no special action. Only at the last second was a clear identification found – a woman with a bike, shopping bags hanging confusingly from handlebars, doubtless assuming the Volvo would route around her as any ordinary vehicle would. Barred from taking evasive action on its own, the computer abruptly handed control back to its human master, but the master wasn't paying attention. Elaine Herzberg, aged 49, was struck and killed, leaving more reflective members of the tech community with two uncomfortable questions: was this algorithmic tragedy inevitable? And how used to such incidents would we, *should* we, be prepared to get?

"In some ways we've lost agency. When programs pass into code and code passes into algorithms and then algorithms start to create new algorithms, it gets farther and farther from human agency. Software is released into a code universe which no one can fully understand."

If these words sound shocking, they should, not least because Ellen Ullman, in addition to having been a distinguished professional programmer since the 1970s, is one of the few people to write revealingly about the process of coding. There's not much she doesn't know about software in the wild.

"People say, 'Well, what about Facebook – they create and use algorithms and they can change them.' But that's not how it works. They set the algorithms off and they learn and change and run themselves. Facebook intervene in their running periodically, but they really don't control them. And particular programs don't just run on their own, they call on libraries, deep operating systems and so on ..."

What is an algorithm?

Few subjects are more constantly or fervidly discussed right now than algorithms. But what *is* an algorithm? In fact, the usage has changed in interesting ways since the rise of the internet – and search engines in particular – in the mid-1990s. At root, an algorithm is a small, simple thing; a rule used to automate the treatment of a piece of data. If *a* happens, then do *b*; if not, then do *c*. This is the "if/then/else" logic of classical computing. If a user claims to be 18, allow them into the website; if not, print "Sorry, you must be 18 to enter". At core, computer programs are bundles of such algorithms. Recipes for treating data. On the micro level, nothing could be simpler. If computers appear to be performing magic, it's because they are fast, not intelligent.

Recent years have seen a more portentous and ambiguous meaning emerge, with the word "algorithm" taken to mean any large, complex decision-making software system; any means of taking an array of input – of data – and assessing it quickly, according to a given set of criteria (or "rules"). This has revolutionized areas of medicine, science, transport, communication, making it easy to understand the utopian view of computing that held sway for many years. Algorithms have made our lives better in myriad ways.

Only since 2016 has a more nuanced consideration of our new algorithmic reality begun to take shape. If we tend to discuss algorithms in almost biblical terms, as independent entities with lives of their own, it's because we have been encouraged to think of them in this way. Corporations like Facebook and Google have sold and defended their algorithms on the promise of objectivity, an ability to weigh a set of conditions with mathematical detachment and absence of fuzzy emotion. No wonder such algorithmic decision-making has spread to the granting of loans/ bail/benefits/college places/job interviews and almost anything requiring choice.

We no longer accept the sales pitch for this type of algorithm so meekly. In her 2016 book Weapons of Math Destruction, Cathy O'Neil, a former math prodigy who left Wall Street to teach and write and run the excellent mathbabe blog, demonstrated beyond question that, far from eradicating human biases, algorithms could magnify and entrench them. After all, software is written by overwhelmingly affluent white and Asian men – and it will inevitably reflect their assumptions (Google "racist soap dispenser" to see how this plays out in even mundane real-world situations). Bias doesn't require malice to become harm, and unlike a human being, we can't easily ask an algorithmic gatekeeper to explain its decision. O'Neil called for "algorithmic audits" of any systems directly affecting the public, a sensible idea that the tech industry will fight tooth and nail, because algorithms are what the companies sell; the last thing they will volunteer is transparency.

The good news is that this battle is under way. The bad news is that it's already looking quaint in relation to what comes next. So much attention has been focused on the distant promises

and threats of artificial intelligence, AI, that almost no one has noticed us moving into a new phase of the algorithmic revolution that could be just as fraught and disorienting – with barely a question asked.

The algorithms flagged by O'Neil and others are opaque but predictable: they do what they've been programmed to do. A skilled coder can in principle examine and challenge their underpinnings. Some of us dream of a citizen army to do this work, similar to the network of amateur astronomers who support professionals in that field. Legislation to enable this seems inevitable.

We might call these algorithms "dumb", in the sense that they're doing their jobs according to parameters defined by humans. The quality of result depends on the thought and skill with which they were programmed. At the other end of the spectrum is the more or less distant dream of human-like artificial general intelligence, or AGI. A properly intelligent machine would be able to question the quality of its own calculations, based on something like our own intuition (which we might think of as a broad accumulation of experience and knowledge). To put this into perspective, Google's DeepMind division has been justly lauded for creating a program capable of mastering arcade games, starting with nothing more than an instruction to aim for the highest possible score. This technique is called "reinforcement learning" and works because a computer can play millions of games quickly in order to learn what generates points. Some call this form of ability "artificial narrow intelligence", but here the word "intelligent" is being used much as Facebook uses "friend" – to imply something safer and better understood than it is. Why? Because the machine has no context for what it's doing and can't do anything else. Neither, crucially, can it transfer knowledge from one game to the next (so-called "transfer learning"), which makes it less generally intelligent than a toddler, or even a cuttlefish. We might as well call an oil derrick or an aphid "intelligent". Computers are already vastly superior to us at certain specialized tasks, but the day they rival our general ability is probably some way off – if it ever happens. Human beings may not be best at much, but we're second-best at an impressive range of things.

Here's the problem. Between the "dumb" fixed algorithms and true AI lies the problematic halfway house we've already entered with scarcely a thought and almost no debate, much less agreement as to aims, ethics, safety, best practice. If the algorithms around us are not yet intelligent, meaning able to independently say "that calculation/course of action doesn't look right: I'll do it again", they are nonetheless starting to learn from their environments. And once an algorithm is learning, we no longer know to any degree of certainty what its rules and parameters are. At which point we can't be certain of how it will interact with other algorithms, the physical world, or us. Where the "dumb" fixed algorithms – complex, opaque and inured to real time monitoring as they can be – are in principle predictable and interrogable, these ones are not. After a time in the wild, we no longer know what they are: they have the potential to become erratic. We might be tempted to call these "frankenalgos" – though Mary Shelley couldn't have made this up.

Clashing codes

These algorithms are not new in themselves. I first encountered them almost five years ago while researching a piece for the Guardian about high frequency trading (HFT) on the stock market. What I found was extraordinary: a human-made digital ecosystem, distributed among racks of black boxes crouched like ninjas in billion-dollar data farms – which is what stock markets had become. Where once there had been a physical trading floor, all action had devolved to a central server, in which nimble, predatory algorithms fed off lumbering institutional ones, tempting them to sell lower and buy higher by fooling them as to the state of the market. Human HFT traders (although no human actively traded any more) called these large, slow participants "whales", and they mostly belonged to mutual and pension funds – ie the public. For most HFT shops, whales were now the main profit source. In essence, these algorithms were trying to outwit each other; they were doing invisible battle at the speed of light, placing and cancelling the same order 10,000 times per second or slamming so many into the system that the whole market shook – all beyond the oversight or control of humans.

No one could be surprised that this situation was unstable. A "flash crash" had occurred in 2010, during which the market went into freefall for five traumatic minutes, then righted itself over another five – for no apparent reason. I travelled to Chicago to see a man named Eric Hunsader, whose prodigious programming skills allowed him to see market data in far more detail than regulators, and he showed me that by 2014, "mini flash crashes" were happening every week. Even he couldn't prove exactly why, but he and his staff had begun to name some of the "algos" they saw, much as crop circle hunters named the formations found in English summer fields, dubbing them "Wild Thing", "Zuma", "The Click" or "Disruptor".

Neil Johnson, a physicist specializing in complexity at George Washington University, made a study of stock market volatility. "It's fascinating," he told me. "I mean, people have talked about the ecology of computer systems for years in a vague sense, in terms of worm viruses and so on. But here's a real working system that we can study. The bigger issue is that we don't know how it's working or what it could give rise to. And the attitude seems to be 'out of sight, out of mind'."

Significantly, Johnson's paper on the subject was published in the journal Nature and described the stock market in terms of "an abrupt system-wide transition from a mixed human-machine phase to a new all-machine phase characterized by frequent black swan [ie highly unusual] events with ultrafast durations". The scenario was complicated, according to the science historian George Dyson, by the fact that some HFT firms were allowing the algos to learn – "just letting the black box try different things, with small amounts of money, and if it works, reinforce those rules. We know that's been done. Then you actually have rules where nobody knows what the rules are: the algorithms create their own rules – you let them evolve the same way nature evolves organisms." Non-finance industry observers began to postulate a catastrophic global "splash crash", while the fastest-growing area of the market

became (and remains) instruments that profit from volatility. In his 2011 novel The Fear Index, Robert Harris imagines the emergence of AGI – of the Singularity, no less – from precisely this digital ooze. To my surprise, no scientist I spoke to would categorically rule out such a possibility.

All of which could be dismissed as high finance arcana, were it not for a simple fact. Wisdom used to hold that technology was adopted first by the porn industry, then by everyone else. But the 21st century's porn is finance, so when I thought I saw signs of HFT-like algorithms causing problems elsewhere, I called Neil Johnson again.

"You're right on point," he told me: a new form of algorithm is moving into the world, which has "the capability to rewrite bits of its own code", at which point it becomes like "a genetic algorithm". He thinks he saw evidence of them on fact-finding forays into Facebook ("I've had my accounts attacked four times," he adds). If so, algorithms are jousting there, and adapting, as on the stock market. "After all, Facebook is just one big algorithm," Johnson says.

"And I think that's exactly the issue Facebook has. They can have simple algorithms to recognize my face in a photo on someone else's page, take the data from my profile and link us together. That's a very simple concrete algorithm. But the question is what is the effect of billions of such algorithms working together at the macro level? You can't predict the learned behavior at the level of the population from microscopic rules. So Facebook would claim that they know exactly what's going on at the micro level, and they'd probably be right. But what happens at the level of the population? That's the issue."

To underscore this point, Johnson and a team of colleagues from the University of Miami and Notre Dame produced a paper, Emergence of Extreme Subpopulations from Common Information and Likely Enhancement from Future Bonding Algorithms, purporting to mathematically prove that attempts to connect people on social media inevitably polarize society as a whole. He thinks Facebook and others should model (or be made to model) the effects of their algorithms in the way climate scientists model climate change or weather patterns.

O'Neil says she consciously excluded this adaptive form of algorithm from Weapons of Math Destruction. In a convoluted algorithmic environment where nothing is clear, apportioning responsibility to particular segments of code becomes extremely difficult. This makes them easier to ignore or dismiss, because they and their precise effects are harder to identify, she explains, before advising that if I want to see them in the wild, I should ask what a flash crash on Amazon might look like.

"I've been looking out for these algorithms, too," she says, "and I'd been thinking: 'Oh, big data hasn't gotten there yet.' But more recently a friend who's a bookseller on Amazon has been telling me how crazy the pricing situation there has become for people like him. Every so often you will see somebody tweet 'Hey, you can buy a luxury yarn on Amazon for

$40,000.' And whenever I hear that kind of thing, I think: 'Ah! That must be the equivalent of a flash crash!'"

Anecdotal evidence of anomalous events on Amazon is plentiful, in the form of threads from bemused sellers, and at least one academic paper from 2016, which claims: "Examples have emerged of cases where competing pieces of algorithmic pricing software interacted in unexpected ways and produced unpredictable prices, as well as cases where algorithms were intentionally designed to implement price fixing." The problem, again, is how to apportion responsibility in a chaotic algorithmic environment where simple cause and effect either doesn't apply or is nearly impossible to trace. As in finance, deniability is baked into the system.

Real-life dangers

Where safety is at stake, this really matters. When a driver ran off the road and was killed in a Toyota Camry after appearing to accelerate wildly for no obvious reason, Nasa experts spent six months examining the millions of lines of code in its operating system, without finding evidence for what the driver's family believed had occurred, but the manufacturer steadfastly denied – that the car had accelerated of its own accord. Only when a pair of embedded software experts spent 20 months digging into the code were they able to prove the family's case, revealing a twisted mass of what programmers call "spaghetti code", full of algorithms that jostled and fought, generating anomalous, unpredictable output. The autonomous cars currently being tested may contain 100m lines of code and, given that no programmer can anticipate all possible circumstances on a real-world road, they have to learn and receive constant updates. How do we avoid clashes in such a fluid code milieu, not least when the algorithms may also have to defend themselves from hackers?

Twenty years ago, George Dyson anticipated much of what is happening today in his classic book Darwin Among the Machines. The problem, he tells me, is that we're building systems that are beyond our intellectual means to control. We believe that if a system is deterministic (acting according to fixed rules, this being the definition of an algorithm) it is predictable – and that what is predictable can be controlled. Both assumptions turn out to be wrong.

"It's proceeding on its own, in little bits and pieces," he says. "What I was obsessed with 20 years ago that has completely taken over the world today are multicellular, metazoan digital organisms, the same way we see in biology, where you have all these pieces of code running on people's iPhones, and collectively it acts like one multicellular organism.

"There's this old law called Ashby's law that says a control system has to be as complex as the system it's controlling, and we're running into that at full speed now, with this huge push to build self-driving cars where the software has to have a complete model of everything, and almost by definition we're not going to understand it. Because any model that we understand is gonna do the thing like run into a fire truck 'cause we forgot to put in the fire truck."

Unlike our old electro-mechanical systems, these new algorithms are also impossible to test exhaustively. Unless and until we have super-intelligent machines to do this for us, we're going to be walking a tightrope.

Federal investigators examine the self-driving Uber vehicle involved in a fatal accident in Tempe, Arizona.

Dyson questions whether we will ever have self-driving cars roaming freely through city streets, while Toby Walsh, a professor of artificial intelligence at the University of New South Wales who wrote his first program at age 13 and ran a tyro computing business by his late teens, explains from a technical perspective why this is.

"No one knows how to write a piece of code to recognize a stop sign. We spent years trying to do that kind of thing in AI – and failed! It was rather stalled by our stupidity, because we weren't smart enough to learn how to break the problem down. You discover when you program that you have to learn how to break the problem down into simple enough parts that each can correspond to a computer instruction [to the machine]. We just don't know how to do that for a very complex problem like identifying a stop sign or translating a sentence from English to Russian – it's beyond our capability. All we know is how to write a more general purpose algorithm that can learn how to do that given enough examples."

Hence the current emphasis on machine learning. We now know that Herzberg, the pedestrian killed by an automated Uber car in Arizona, died because the algorithms wavered in correctly categorizing her. Was this a result of poor programming, insufficient algorithmic training or a hubristic refusal to appreciate the limits of our technology? The real problem is that we may never know.

"And we will eventually give up writing algorithms altogether," Walsh continues, "because the machines will be able to do it far better than we ever could. Software engineering is in that sense perhaps a dying profession. It's going to be taken over by machines that will be far better at doing it than we are."

Walsh believes this makes it more, not less, important that the public learn about programming, because the more alienated we become from it, the more it seems like magic beyond our ability to affect. When shown the definition of "algorithm" given earlier in this piece, he found it incomplete, commenting: "I would suggest the problem is that algorithm now means any large, complex decision making software system *and* the larger environment in which it is embedded, which makes them even more unpredictable." A chilling thought indeed. Accordingly, he believes ethics to be the new frontier in tech, foreseeing "a golden age for philosophy" – a view with which Eugene Spafford of Purdue University, a cybersecurity expert, concurs.

"Where there are choices to be made, that's where ethics comes in. And we tend to want to have an agency that we can interrogate or blame, which is very difficult to do with an

algorithm. This is one of the criticisms of these systems so far, in that it's not possible to go back and analyze exactly why some decisions are made, because the internal number of choices is so large that how we got to that point may not be something we can ever recreateto prove culpability beyond doubt."

The counter-argument is that, once a program has slipped up, the entire population of programs can be rewritten or updated so it doesn't happen again – unlike humans, whose propensity to repeat mistakes will doubtless fascinate intelligent machines of the future. Nonetheless, while automation should be safer in the long run, our existing system of tort law, which requires proof of intention or negligence, will need to be rethought. A dog is not held legally responsible for biting you; its owner might be, but only if the dog's action is thought foreseeable. In an algorithmic environment, many unexpected outcomes may not have been foreseeable to humans – a feature with the potential to become a scoundrel's charter, in which deliberate obfuscation becomes at once easier and more rewarding. Pharmaceutical companies have benefited from the cover of complexity for years (see the case of Thalidomide), but here the consequences could be both greater and harder to reverse.

[…]

Searching for a solution

'Basically, we need a new science,' says Neil Johnson.

Solutions exist or can be found for most of the problems described here, but not without incentivizing big tech to place the health of society on a par with their bottom lines. More serious in the long term is growing conjecture that current programming methods are no longer fit for purpose given the size, complexity and interdependency of the algorithmic systems we increasingly rely on. One solution, employed by the Federal Aviation Authority in relation to commercial aviation, is to log and assess the content of all programs and subsequent updates to such a level of detail that algorithmic interactions are well understood in advance – but this is impractical on a large scale. Portions of the aerospace industry employ a relatively new approach called model-based programming, in which machines do most of the coding work and are able to test as they go.

Model-based programming may not be the panacea some hope for, however. Not only does it push humans yet further from the process, but Johnson, the physicist, conducted a study for the Department of Defense that found "extreme behaviors that couldn't be deduced from the code itself" even in large, complex systems built using this technique. Much energy is being directed at finding ways to trace unexpected algorithmic behavior back to the specific lines of code that caused it. No one knows if a solution (or solutions) will be found, but none are likely to work where aggressive algos are designed to clash and/or adapt.

As we wait for a technological answer to the problem of soaring algorithmic entanglement, there are precautions we can take. Paul Wilmott, a British expert in quantitative analysis and vocal critic of high frequency trading on the stock market, wryly suggests "learning to shoot, make jam and knit". More practically, Spafford, the software security expert, advises making tech companies responsible for the actions of their products, whether specific lines of rogue code – or proof of negligence in relation to them – can be identified or not. He notes that the venerable Association for Computing Machinery has updated its code of ethics along the lines of medicine's Hippocratic oath, to instruct computing professionals to do no harm and consider the wider impacts of their work. Johnson, for his part, considers our algorithmic discomfort to be at least partly conceptual; growing pains in a new realm of human experience. He laughs in noting that when he and I last spoke about this stuff a few short years ago, my questions were niche concerns, restricted to a few people who pored over the stock market in unseemly detail.

"And now, here we are – it's even affecting elections. I mean, what the heck is going on? I think the deep scientific thing is that software engineers are trained to write programs to do things that *optimize* – and with good reason, because you're often optimizing in relation to things like the weight distribution in a plane, or a most fuel-efficient speed: in the usual, anticipated circumstances optimizing makes sense. But in unusual circumstances it doesn't, and we need to ask: 'What's the worst thing that could happen in this algorithm once it starts interacting with others?' The problem is we don't even have a word for this concept, much less a science to study it."

He pauses for moment, trying to wrap his brain around the problem.

"The thing is, optimizing is all about either maximizing or minimizing something, which in computer terms are the same. So what *is* the opposite of an optimization, ie the least optimal case, and how do we identify and measure it? The question we need to ask, which we never do, is: 'What's the most extreme possible behavior in a system I thought I was optimizing?'"

Another brief silence ends with a hint of surprise in his voice.

"Basically, we need a new science," he says.

https://www.theguardian.com/technology/2018/aug/29/coding-algorithms-frankenalgos-program-danger

Accessed: 11 September 2018